Regularization, Adaptation and Generalization of Neural Networks

Author: Riccardo Volpi

Advisor: Prof. Vittorio Murino

Istituto Italiano di Tecnologia

Università degli Studi di Genova

February, 2019

The ability to generalize to unseen data is one of the fundamental, desired properties in a learning system. This thesis reports different research efforts in improving the generalization properties of machine learning systems at different levels, focusing on neural networks for computer vision tasks.

First, a novel regularization method is presented, *Curriculum Dropout*. It combines Curriculum Learning and Dropout, and shows better regularization effects than the original algorithm in a variety of tasks, without requiring substantially any additional implementation efforts.

While regularization methods are extremely powerful to better generalize to unseen data from the *same* distribution as the training one, they are not very successful in mitigating the *dataset bias* issue. This problem constitutes in models learning the peculiarities of the training set, and poorly generalizing to unseen domains. Unsupervised domain adaptation has been one of the main solutions to this problem. Two novel adaptation approaches are presented in this thesis. First, we introduce the *DIFA* algorithm, which combines domain invariance and feature augmentation to better adapt models to new domains by relying on adversarial training. Next, we propose an original procedure that exploits the "mode collapse" behavior of Generative Adversarial Networks.

Finally, the general applicability of domain adaptation algorithms is questioned (due to the assumptions of knowing the target distribution a priori and being able to sample from it). A novel framework is presented to overcome its liabilities, where the goal is to generalize to unseen domains by relying only on data from a *single* source distribution. We face this problem through the lens of robust statistics, defining a worst-case formulation where the model parameters are optimized with respect to populations which are ρ-distant from the source domain on a semantic space.

Contents

Acronyms

- **ADDA:** Adversarial Discriminative Domain Adaptation
- **BN:** Batch Normalization
- **BSA:** Bi-Shifting Auto-Encoder
- **cGAN:** conditional Generative Adversarial Networks
- **CoGAN:** Coupled Generative Adversarial Networks
- **CNN:** Convolutional Neural Network
- **DDC:** Deep Domain Confusion
- **DIFA:** Domain Invariance - Feature Augmentation
- **DTN:** Domain Transfer Network
- **DANN:** Domain Adversarial Neural Network
- **ERM:** Empirical Risk Minimization
- **FC:** Fully Connected
- **FCN:** Fully Convolutional Network
- **GAN:** Generative Adversarial Networks

- **GDSDA:** Generalized Distillation Semi-supervised Domain Adaptation
- **GM:** Gaussian Mixture
- **KL:** Kullback–Leibler
- **LS:** Least-Squares
- **MNIST:** Modified National Institute of Standards and Technology
- **SDASL:** Semi-supervised Domain Adaptation with Subspace Learning
- **SVHN:** Street View House Numbers
- **UNIT:** UNsupervised Image-to-image Translation
- **VAE:** Variational Auto-Encoder
- **XE:** Cross-Entropy

Chapter 1

Introduction

1.1 Overview

In the last few years, deep learning systems have become ubiquitous in the broad area of computer vision. Deep convolutional neural networks allow to achieve remarkable results in object recognition [79, 138, 55, 64], semantic segmentation [6, 123, 88] and object detection [117, 118, 119]. However, the outstanding performance of this class of algorithms comes with a price: generally, these models are characterized by a very large number of parameters (typically in the order of 10^7), and the training procedure requires a huge number of annotated samples. Due to the large number of parameters, these models are prone to overfitting, and different strategies need to be adopted to make them generalize to unseen data with good performances.

When we discuss generalization properties, a distinction needs to be made: whether the aim is to generalize to unseen data from the *same* distribution as the training one, or the aim is to generalize to data from *different* distributions. In the former case, the problem can be faced through regularization techniques, that have been studied for several decades. In the latter case, the problem is

different: due to the inherent bias that typically characterizes each dataset [152], models trained on data from some distribution will poorly generalize to samples drawn from distributions different than the training one. In this case, different approaches can be adopted, the two more widely known being domain adaptation and domain generalization.

Regularization

The problem of generalizing to new data from the same distribution as the training one is well studied in machine learning, and can be approached through regularization techniques.

Consider a classification problem, where we desire to learn a function $f_\theta(x)$, where x is a datapoint (*e.g.*, an image) and the output of this function is the class $\tilde{y}$ the datapoint is estimated to belong to. If we have a dataset $\{x^{(i)}, y^{(i)}\}_{i=1...N}$, one of the simplest approach to learn the parameters θ that define the function is Empirical Risk Minimization (ERM)

$$\min_\theta \hat{\varepsilon}(\theta) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y^{(i)}, f_\theta(x^{(i)})) \tag{1.1}$$

where we aim at minimizing the risk $\hat{\varepsilon}$, associated with a loss $\mathcal{L}$ that measures the discrepancy between the output of our function and the ground truth. The issue with ERM is overfitting the training set: given a model θ with sufficient capacity, we can reduce our risk to 0, but poorly generalizing to new datapoints x^{new}. This is a particularly concrete problem if our dataset size is limited with respect to the capacity of the model (intuitively, if N is small and θ is big).

Regularization approaches such as ridge (Tikhonov regularization, [150]) and lasso [149] aim at solving this problem. The core idea is to penalize our algorithm to learn complex models, by adding a regularization term

$$\min_{\theta} \mathcal{F}(\theta) \coloneqq \hat{\varepsilon}(\theta) + \lambda R(\theta) \tag{1.2}$$

where we can have, *e.g.*, $R = \|\theta\|_2^2$ for ridge (also referred to as *weight decay*) or $R = \|\theta\|_1$ for lasso.

A different, widely used approach is constituted by *early stopping* [106, 37], where the training procedure is stopped before the model learned is too complex to perform well on new datapoints.

The hyperparameters that typically characterize regularization methods (*e.g.*, λ in Equation 1.2) and how early the training procedure should be stopped can be estimated through validation on a set of samples that is held from the training set. These well-known approaches can be applied also in deep learning, where, as detailed above, we are using models with extremely high capacity (millions of parameters).

Concerning deep neural networks though, recent techniques designed *ad hoc* for this class of models proved to perform better than "standard" regularization approaches: Dropout [141] and Batch Normalization [68].

Dropout [141] was one of the key characteristics of the AlexNet model introduced by Krizhevsky et al. [79], where the authors claim that "without dropout, network exhibits substantial overfitting". The core idea is to drop each unit with probability p throughout the training procedure, avoiding in this way the co-adaptation between feature detectors. Batch Normalization (BN) [68] is a key part of modern deep architectures [55, 64]. The idea behind BN is to normalize the output of each layer, and this results in faster convergence rates and better generalization properties. Both Dropout [141] and BN [68] methods are detailed in Chapter 2.

The approaches described in this paragraph have only limited success when the training distribution differs from the distribution on which the

learning system is deployed. The following paragraphs describe different generalization scenarios, as well as possible solutions.

The dataset bias issue

As mentioned, if the goal if to devise learning systems able to generalize to samples from distributions different than the training one, standard regularization methods are not very effective. In this scenario, we have to face a different problem: each dataset contains its own bias [152], that can be defined as the set of characteristics that makes each dataset unique. This bias is detrimental while training models, because they learn to rely on it during training, which, in turn, results in models that overfit on the data population from which training samples are drawn, and thus poorly generalize on samples from different, unseen domains [67, 14, 8, 128, 152].

For example, consider a vision module for self-driving cars that is trained using only data from sunny, daylight scenarios. The module will poorly generalize, *e.g.*, to dark or foggy scenarios. The same example could be extended in the case where training samples come from a single city. The module will be biased towards the architecture and the types of street as they are in the training set, and will poorly generalize to other cities. The best option would be to have a training set that comprises samples from the broad set of domains on which the model will be deployed, but naturally this scenario is rarely an option, in particular due to the expensiveness of obtaining ground truth for data.

Figure 1.1 shows a simple example of dataset bias. While MNIST dataset (right, [82]) is one of the easiest dataset in computer vision (very simple models achieve an accuracy $> 99\%$ on the test set), if we train a model on SVHN (left, [111]) we perform poorly on samples drawn from MNIST.

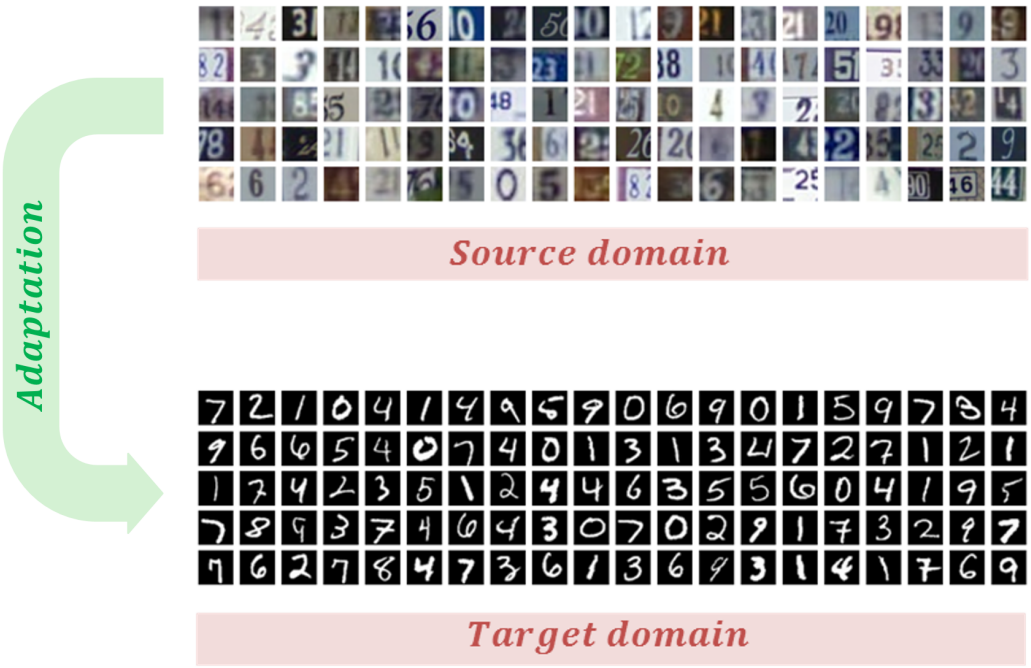


Figure 1.1. The SVHN (source) → MNIST (target) split, a benchmark experiment in unsupervised domain adaptation.

To overcome the lack of generalization on out-of-distribution samples, the *domain adaptation* and *domain generalization* frameworks have been proposed

Domain adaptation. The basic idea of domain adaptation is that, during training, one can exploit, in addition to the labeled samples from the (source) training distribution, also a set of samples from a desired (target) distribution, with limited or missing annotation.

More formally, in *supervised* domain adaptation [67, 66] the training data is represented by two sets of samples $\{x^{(i)}, y^{(i)}\}_{1...m} \sim P_{source}$ and $\{x^{(i)}, y^{(i)}\}_{1...p} \sim P_{target}$, typically with $p \ll m$, where P_{source} and P_{target} are the source and the target distributions, respectively. In *semi-supervised* domain adaptation [14, 26], we also rely on a set of unlabeled data from the target distribution, $\{x^{(i)}\}_{1...n} \sim P_{target}$. In *unsupervised* domain adaptation

[128, 36], we only rely on labeled samples from the source distribution and unlabeled samples from the target distribution.

Concerning domain adaptation, in this thesis we only focus on the unsupervised setting. Different approaches allow to solve this problem efficiently in a plethora of tasks. Adversarial training has been effectively used to map source and target samples in a common feature space [39, 40, 153, 158]. Other works aim at aligning the second order statistics of source and target features [147, 104]. More recently, several methods that rely on image-to-image translation methods to learn the mapping from the source space to the target one and vice-versa have been proposed [90, 148, 127, 16, 89, 132, 61]. In general, one can design models for unsupervised domain adaptation that leverage labeled source samples that are "rendered" with the style of target samples, and vice-versa. These are a few examples of how to face the task; in Chapter 2, a detailed description of the state of the art is reported.

Beyond adaptation. While the effectiveness of unsupervised domain adaptation algorithms has steadily grown in the last years, this framework is based on two strong assumptions: it is necessary (i) to define the target distribution a priori and (ii) to be able to sample unlabeled data from it. While the second assumption is typically realistic (obtaining unlabeled samples is cheap), the first one can be a limitation.

For example, consider the self-driving car example presented above. In order to adapt the system to the broad spectrum of different conditions, one has to know a priori where the driver will use his car and in which conditions. As a second example, consider a semantic segmentation algorithm used by a robot: every task, robot, environment and camera configuration will result in a different target distribution, and these diverse scenarios can be identified

only after the model is trained and deployed, making it difficult to collect samples from them.

What we would ideally desire from a learning system is the ability to *generalize* to new domains. A possible way to face this problem is by following the approaches from the *domain generalization* field, where the aim is to learn models that better generalize to unseen domains by leveraging data from different source populations (for example, see [74, 165, 107, 41, 85, 134, 96, 98, 86]). The limitation of these algorithms is the assumption that training data is defined by different populations, and indeed typically also require data in the form $\{x^{(i)}, y^{(i)}, d^{(i)}\}_{1...m} \sim P_{source}$, where d is the label of the sub-domain a datapoint belongs to.

1.2 Contributions

In this section, the contributions brought by this thesis are introduced. They represent attempts to deal with the limitations of neural networks detailed in this Chapter. First, a novel way to perform Dropout [141] is introduced; next, two different algorithms for unsupervised domain adaptation are presented; finally, a novel approach to cope with single-source domain generalization is defined.

A novel approach to dropout training

In this thesis, we propose a novel dropout training procedure, termed *Curriculum Dropout* [105]. As the name suggests, it combines Dropout [141] and Curriculum Learning [9].

We have already mentioned that Dropout prevents overfitting by randomly "dropping" some of the neural network units during training. Curriculum

Learning [9] is a training paradigm based on the idea that simple concepts should be learned before the more complex ones, drawing inspiration from the way humans learn.

We draw inspiration from the latter concept, posing the question on whether dropping units with a consistent probability p throughout the Dropout [141] training procedure is an optimal solution. Indeed, by suppressing units we make the training procedure more difficult for the model (it is a well known fact that training with Dropout requires more iterations [141]).

We propose a scheduling for the dropout rate, where, at the beginning of the training procedure, the probability p to drop a unit is 0, and then it increases over iterations. This is in line with the *start easy* philosophy that characterizes Curriculum Learning [9]. The main idea behind the proposed method is that overfitting is typically faced later in training. For this reason, we argue that applying Dropout [141] at the beginning of the training procedure might not be an optimal procedure. We define a formal connection between our method and Curriculum Learning [9] and show that Curriculum Dropout leads to better accuracy than the original algorithm [141] in a variety of tasks, without requiring any substantial, additional computational effort.

Novel approaches to unsupervised domain adaptation

In this thesis, we propose two novel algorithms for unsupervised domain adaptation. In the previous section, we had defined two different ways to exploit adversarial training in unsupervised domain adaptation: (i) for source/target feature confusion and (ii) for image-to-image translation. We propose two contributions, one from the former class (i) [158] and one intimately related with the latter (ii).

The DIFA algorithm. We build on the work by Tzeng et al. [153], where the authors propose a training procedure that uses an objective inspired by Generative Adversarial Networks (GANs [45]) to learn target features that are indistinguishable from the source ones, resulting in a couple of feature extractors, one for the source samples and one for the target ones.

We extend this approach in two directions: (a) we force domain-invariance in a single feature extractor trained through a GAN objective, and (b) we perform data augmentation in the feature space (i.e., *feature augmentation*), by defining a more complex minimax game. We perform feature augmentation by introducing the use of conditional GANs (cGANs, [102]) to model feature distributions. By modeling the source feature distribution with a generative model, we can generate an arbitrary number of labeled feature vectors, by conditioning on the desired classes. Our results show that forcing domain-invariance and augmenting features are both valuable approaches in the unsupervised domain adaptation setting. We term the proposed method *DIFA*, which stands for "Domain Invariance - Feature Augmentation".

Exploiting GAN mode collapse. We study the behavior of cGANs when training data is polluted with label noise (namely, an unknown percentage of training samples is provided with the wrong label). We observe that, when label pollution is reasonably below some percentage, cGANs are robust against this kind of noise, and allow to generate "cleaner" data samples than the ones from the original distribution. This is the result of GANs collapsing into regions of data with higher probability (mode collapse [47]).

A natural application of these findings is the generation of less polluted datasets, namely datasets with a lower amount of label noise. In light of this, we propose unsupervised domain adaptation as an application, approaching it

as a *noisy label* problem. More in detail, if we have a model $M_{\theta_s}(x; \theta_s)$ whose weights θ_s are trained on labeled samples from the source domain, we can infer a pseudo-label for each target sample, obtaining a label-polluted target set $\{x_t^{(i)}, \tilde{y}_t^{(i)}\}_{i...M}$, where typically a consistent number of $\tilde{y}_t^{(i)}$ are mistaken.

We propose a training procedure where a cGAN is trained on the noisy target set and a classifier is trained on the "cleaner" generated data. The idea is that the classifier benefits from the more reliable data generated, and the cGAN benefits from the more reliable pseudo-labels inferred by the classifier. In this framework, the source samples are thus exploited only to train an initial classifier M_{θ_s}. After this step, the problem is faced in a fully unsupervised fashion where the noise on the labels of the empirical target distribution is reduced over iterations.

A novel view on domain generalization

To overcome the limitations of domain adaptation and generalization we mentioned, in this thesis we introduce a new framework for generalization. In the proposed setting, a learning system needs to generalize to unknown domains by relying only on data from a *single* source distribution during training [159].

We face this task through the lens of robust statistics, by defining a worst-case formulation where a set of distributions which are close to the source one in terms of semantic distance are considered. Intuitively, we generate samples from fictitious distributions that are difficult for the current model, and optimize the model parameters with respect to those samples. We propose to solve the problem through a data augmentation pipeline, where new, adversarial samples are appended to the original dataset at each iteration.

From a practical viewpoint, a key difficulty in applying the worst-case formulation is that the magnitude of the semantic distance is a priori unknown. We propose to learn an ensemble of models that correspond to different distances. In other words, our iterative method generates a collection of datasets, each corresponding to a different inter-dataset distance level, and we learn a model for each of them. At test time, we use a heuristic method to choose an appropriate model from the ensemble.

Results in cross-dataset object recognition and cross-weather/city semantic segmentation benchmarks show that our method systematically outperforms both models trained via Empirical Risk Minimization and models regularized with standard regularization techniques (ridge [150], Dropout [141]). We also show that, in some scenarios, our method performs comparably with unsupervised domain adaptation algorithms, even if they make use of additional data and have prior knowledge on the target domain.

1.3 Publications

We report in the following the list of publications and submissions related to the content of various Chapters.

1. Pietro Morerio, Jacopo Cavazza, **Riccardo Volpi**, René Vidal, Vittorio Murino, *Curriculum Dropout*, The IEEE International Conference on Computer Vision (ICCV), 2017, Venice, Italy. (Chapter 3).

2. **Riccardo Volpi**, Pietro Morerio, Silvio Savarese, Vittorio Murino, *Adversarial Feature Augmentation for Unsupervised Domain Adaptation*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, Salt Lake City, Utah. (Chapter 4).

3. **Riccardo Volpi***, Hongseok Namkoong*, Ozan Sener, John Duchi, Vittorio Murino, Silvio Savarese, *Generalizing to Unseen Domains via Adversarial Data Augmentation*, 32^{nd} Conference on Neural Information Processing Systems (NIPS), 2018, Montreal, Canada. (Chapter 6).

4. Pietro Morerio*, **Riccardo Volpi***, Vittorio Murino, *The Bright Side of Mode Collapse: Pseudo-label Refinement for Unsupervised Domain Adaptation*, currently under review. (Chapter 5).

1.4 Summary

The rest of this thesis is organized as follows. In Chapter 2, background and related work are detailed. In Chapter 3, the *Curriculum Dropout* training procedure is presented and analyzed through several experiments on benchmark datasets. In Chapters 4 and Chapter 5, respectively, the *DIFA* algorithm and the iterative procedure based on mode collapse are described and evaluated on unsupervised domain adaptation benchmarks. In Chapter 6, the new framework to study generalization is presented, and the adversarial data augmentation algorithm is empirically and theoretically analyzed. In Chapter 7, we draw the final remarks.

Chapter 2

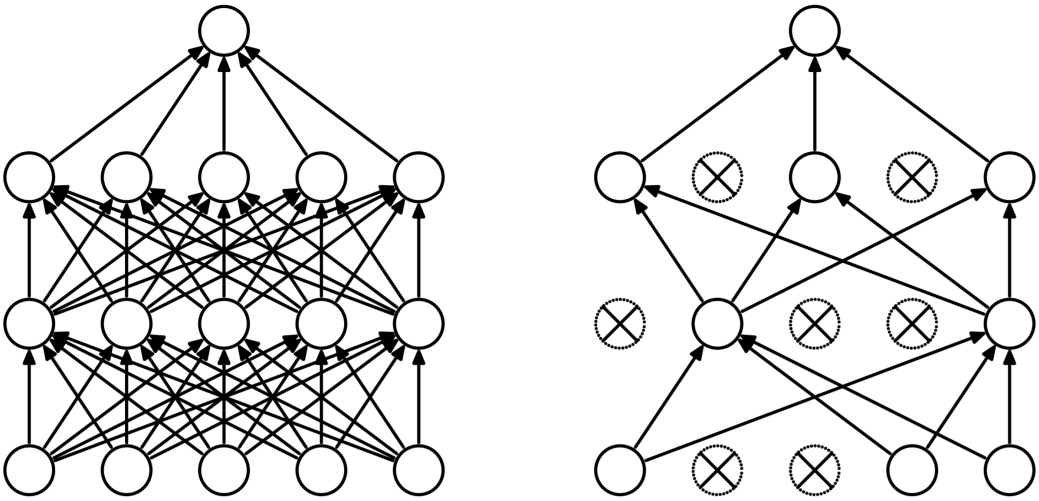
Background and related work

In this chapter, we detail the main approaches used to regularize deep neural networks (Dropout [141] and BN [68]), the state of the art of domain adaptation and domain generalization, and the details required to a basic understanding of GANs [45].

2.1 Regularization techniques

In the previous Chapter, we had discussed regularization techniques for machine learning systems in general, from standard solutions such as ridge [150], lasso [149] or early stopping [106, 37], to more effective solutions for deep learning models. We had defined the key strategies to prevent overfitting in deep networks as Dropout [141] and BN [68].

In the following sections, we cover these topics in more details. The literature related to dropout training is naturally important for this thesis, being Chapter 3 dedicated to a novel way to perform dropout, thus the standard algorithm [141] and its variants are detailed in the following. BN [68] is not strictly related to this thesis, but we cover the basic details of the



(a) Standard Neural Net (b) After applying dropout

Figure 2.1. Differences between network architecture without Dropout [141] *(a)* and with Dropout [141] *(b)* during the training procedure. As can be observed in *(b)*, when dropout training is performed, some units are "dropped" (crosses) with probability p. Figure from [141].

approach, for the sake of completeness.

2.1.1 Dropout training

The *Dropout* training procedure is introduced by Hinton et al. [60] and Sivrastava et al. [142]. Dropout training consists in randomly "dropping" units during the training procedure of a neural network, in order to prevent co-adaptation between feature detectors.

Intuitively, this results in 2^N different configurations, where N is the number of units, and we can interpret this as an exponentially large ensemble. During test, the weights of each unit are multiplied by the probability of that unit to be active during the training phase. In the seminal works [60, 142], the method is detailed and evaluated with different types of deep learning models (Multi-Layer Perceptrons, Convolutional Neural Networks,

Restricted Boltzmann Machines) and datasets, confirming the effectiveness of this approach against overfitting.

After the publication of the original algorithm, many works [162, 87, 163, 7, 164, 69, 161, 121] have investigated the topic. Wan et al. [162] propose Drop-Connect, a more general version of Dropout. Instead of directly setting units to zero, only some of the network connections are suppressed. This generalization is proven to be better in performance but slower to train with respect to [60, 142]. Li et al. [87] introduce data-dependent and Evolutional-dropout for shallow and deep learning, respectively. These versions are based on sampling neurons form a multinomial distribution with different probabilities for different units. Results show faster training and sometimes better accuracies. Wang et al. [163] accelerate dropout. In their method, hidden units are dropped out using approximated sampling from a Gaussian distribution. Results show that the approach leads to faster convergence without deteriorating the accuracy. Bayer et al. [7] carry out a fine analysis, showing that dropout can be proficiently applied to Recurrent Neural Networks. Wu and Gu [164] analyze the effect of dropout on the convolutional layers of a Convolutional Neural Networks (CNN, [80]). They define a probabilistic weighted pooling, which effectively acts as a regularizer. Zhai and Zhang [168] investigate the idea of dropout once applied to matrix factorization. Ba and Frey [69] introduce a binary belief network which is overlaid on a neural network to selectively suppress hidden units. The two networks are jointly trained, making the overall process more computationally expensive. Wager et al. [161] apply Dropout on generalized linear models and approximately prove the equivalence between data-dependent L^2 regularization and dropout training with AdaGrad optimizer. Rennie et al. [121] propose to adjust the dropout rate, linearly decreasing the unit suppression rate during training,

Algorithm 1 Batch Normalization

Input 1 mini-batch of activations $B = \{x_i\}_{i=1,...,n}$

Input 2 parameters to be learned γ, β

Output $\{y_i\}_{i=1,...,n}$, with $y_i = BN_{\gamma,\beta}(x_i)$

$\mu_B \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i$

$\sigma_B \leftarrow \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_B)^2$

$\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sigma_B + \epsilon}$

$y_i \leftarrow \gamma\hat{x_i} + \beta$

until the network experiences no dropout.

2.1.2 Batch normalization

BN [68] is one of the key features of modern deep neural networks [55], allowing to (i) accelerate the training procedure by using larger learning rates and (ii) carrying a regularization benefit. The general concept that BN [68] is based on is that the training procedure of neural networks benefits from input normalization [80]. BN [68] extends this concept to each activation layer of a neural network. Algorithm 1 summarizes the four steps that characterize the application of BN [68]. The parameters γ and β are scaling and shifting coefficients, respectively, that are learned during the training procedure.

2.2 Domain adaptation

In the previous Chapter, we had defined the adaptation goal as the aim to train models that perform well on a data distribution of interest (target) $P_{target}(X)$, for which samples are provided in different forms.

In particular, we had defined three different scenarios, summarized in the following:

- **Supervised domain adaptation.** During training, we are provided with a set of labeled samples from a source distribution, $\{x^{(i)}, y^{(i)}\}_{1...m} \sim P_{source}(X, Y)$, and a set of labeled samples from the target distribution, $\{x^{(i)}, y^{(i)}\}_{1...p} \sim P_{target}(X, Y)$, typically with $p \ll m$.

- **Semi-supervised domain adaptation.** During training, we are provided with a set of labeled samples from a source distribution, $\{x^{(i)}, y^{(i)}\}_{1...m} \sim P_{source}(X, Y)$, a set of labeled samples from the target distribution, $\{x^{(i)}, y^{(i)}\}_{1...p} \sim P_{target}(X, Y)$, typically with $p \ll m$, and a set of unlabeled samples from the target distribution, $\{x^{(i)}\}_{1...n} \sim P_{target}(X)$.

- **Unsupervised domain adaptation.** During training, we are provided with a set of labeled samples from a source distribution, $\{x^{(i)}, y^{(i)}\}_{1...m} \sim P_{source}(X, Y)$, and a set of unlabeled samples from the target distribution, $\{x^{(i)}\}_{1...n} \sim P_{target}(X)$.

This thesis focuses on the third, more general scenario. In the following section, the state of the art is covered in details. Also a section dedicated to the state of the art of semi-supervised domain adaptation is included, for the sake of completeness.

2.2.1 State of the art: approaches and methods

Unsupervised domain adaptation. Glorot et al. [42] propose to use autoencoders to learn features representative for both source and target samples. Kan et al. [71] introduce a bi-shifting autoencoder (BSA), used to shift source domain samples into target ones. Different works [63, 135] approach the problem relying on dictionary learning, in order to find representations where source and target samples are aligned. Geodesic methods [48, 43] aim

at connecting source and target samples with a path on a manifold, on which samples from both source and target domains can be projected. Other works [146, 36] introduce transformations to minimize the distance between source and target covariances.

In recent years, the application of deep neural networks for domain adaptation has been investigated, to reduce the domain shift via end-to-end training. Chopra et al. [23] introduce a method where an interpolating path between the two domains is defined, and features are extracted in an unsupervised fashion from the interpolating domains. Such features are further combined and filled to a classifier. Tzeng et al. [154] define an adaptation layer and an additional domain adaptation loss that are embedded in a standard CNN architecture. Ganin and Lempitsky [39] propose a neural network (Domain-Adversarial Neural Network, DANN) where a CNN-based [82] feature extractor is optimized to both correctly classify source samples and have domain-invariant features, through adversarial training. In Long et al. [92], hidden representations are explicitly matched in Reproducing Kernel Hilbert Spaces. In Rozantsev et al. [125], a two stream neural network is proposed, where one is dedicated to modeling the source samples and the other to modeling the target ones, without weight sharing. In Sener et al. [133], the representation, the transformation between the two domains and the target label inference are optimized in a end-to-end fashion, exploiting transductive learning. Different works [155, 93] aim at minimizing the Maximum Mean Discrepancy [49] between features extracted from source and target samples, training a classifier to correctly classify source samples while minimizing this measure. Bousmalis et al. [17] propose to learn image representations divided into two components, one shared across domains and one private, following the hypothesis that modeling unique elements in each domain can help to extract

features which are domain-invariant. Tzeng et al. [153] use GANs to train an encoder for target samples, by making the features extracted with this model indistinguishable from the ones extracted through an encoder trained with source samples. The last layer of the latter can then be used for both encoders to infer labels. Saito et al. [129] propose an asymmetric tri-training procedure where pseudo-labels are inferred and exploited for target samples during training. In particular, two networks are trained to assign labels to target samples and one to obtain target-discriminative features. Haeusser et al. [53] propose to exploit associations between source and target features during training, to maximize the domain-invariance of the learned features while minimizing the error on source samples. Sun and Saenko [147] and Morerio et al. [104] propose an end-to-end training procedure where the second-order statistics of source and target features is aligned, while the network is simultaneously trained to solve the desired task. Saito et al. [130] propose an adversarial procedure that optimizes for having target samples far from the decision boundaries, encouraging the generator to output more discriminative features. Shu et al. [136] address some potential issues of adversarial training for unsupervised domain adaptation by following the "cluster assumption", namely assuming that decision boundaries should not cross high-density data regions, and propose two different algorithms (VADA and DIRT-T). French et al. [38] introduce the use of self-ensembling for domain adaptation. Mancini et al. [98] make the assumption the the source domain is a mixture of different data populations, and propose a solution to extract information regarding the different sub-domains and exploit this information to better adapt to the target domain. Saito et al. [131] introduce a method to detect target samples that are far from the support of the source, and propose a feature generator that learns to generate target features near

the support to minimize the discrepancy. Pinheiro [114] proposes to exploit similarity learning for the unsupervised domain adaptation problem, defining a method based on a pairwise similarity function, where classification is performed through the computation of the similarity with categorical prototype representations. Kang et al. [72] introduce the use of attention in unsupervised domain adaptation, proposing an "attention alignment scheme", following the assumption that the discriminative regions of the images are the same for both source and target samples; the authors also propose a way to estimate the posterior distribution of the target labels. Damodaran et al. [11] propose to face the unsupervised domain adaptation problem through the lens of optimal transport, introducing DeepJDOT, which allows to both reduce the discrepancy between source/target distributions while preserving discriminative information used by the classifier.

Several image-to-image translation methods have been proposed in recent years, benefiting from the success of Generative Adversarial Networks [45]. Other than natural application in graphics, those algorithms can be used to face unsupervised domain adaptation tasks. Taigman et al. [148] propose the Domain Transfer Network (DTN), that allows to translate images from a source domain to a target one, under a f-constancy constraint, where f is a generic function that maps images in a feature space. Translated images result portrayed in the target images' style, while maintaining the content of the images fed in input. Liu and Tuzel [90] introduce Coupled GAN (CoGAN), an extension of GAN that allows to model a joint distribution $P(A, B)$ and to generate couples of images from noise vectors, one belonging to $P(A)$ and one to $P(B)$. This model can be applied to image-to-image translation tasks: fixing one image, the noise vector that most likely could have generated that picture can be inferred and, feeding it to the model,

the second image is generated. Bousmalis et al. [16] propose to train an image-to-image translation network relying on both a GAN loss and a *task-specific* loss (and in problems with prior knowledge, also a *content-specific* loss). The resulting network takes in input both an image and a noise vector, that allows to generate a potentially infinite number of target images. Liu et al. [89] propose UNIT, an extension of CoGAN that relies on both GANs and Variational Auto-Encoders (VAE, [76]), and makes the assumption of a shared latent space. Russo et al. [127] exploit GANs in unsupervised domain adaptation by defining a symmetric mapping between source and target domains, imposing to preserve the class identity of the transformed images. Murez et al. [108] propose an image-to-image translation method where it is imposed that the features extracted by an encoder network are able to reconstruct the images in both domains. Generally, image-to-image translation methods can be applied to the unsupervised domain adaptation problem by converting target images into a style that resembles the one of the source domain, or vice-versa.

Semi-supervised domain adaptation. Daume III et al. [26] propose $EA++$, an extension of an existing method for supervised domain adaptation (EA [26]) that is based on the idea of augmenting the source feature space with features from target samples. $EA++$ generalizes this approach by also leveraging on unlabeled data from the target domain. Lopez et al. [94] propose a density model able to adapt across different distributions, based on the theory of copulas. Donahue et al. [29] force smoothness constraints on the output of a classifier over the unlabeled data from the target domain, and successfully test this idea on top of a number of approaches that only leverage on labeled samples. Yao et al. [166] propose the SDASL method

(Semi-supervised Domain Adaptation with Subspace Learning), which, at the same time, explores domain-invariant features to correct the shift between the distributions and leverages on unlabeled target samples to exploit the underlying strusture of the target distribution. Ao et al. [2] introduce the GDSDA method (Generalized Distillation Semi-supervised Domain Adaptation), where knowledge from source models is transferred to target models through distillation.

2.3 Domain generalization

In recent years, the domain generalization task has been proposed [74, 107]. The goal is to generalize to unseen domains, by relying on data from different source populations, usually in the form $\{x_i, y_i, d_i\} \sim P_{source}(X, Y, D)$, where $P_{source}(X, Y, D)$ is the joint distribution of source samples, their class labels and their domain labels.

In the following section, the body of works [74, 165, 107, 41, 85, 134, 96, 98, 86] that address this task is detailed. The different works detailed in the following require to know the labels related to the domain each sample belongs to during training. Mancini et al. [96] is an exception as the authors also propose a method to automatically infer the domain labels, but also this work makes the assumption that the source domain comprises different sub-domains.

In this thesis the domain generalization task is not solved in this form. The approach detailed in Chapter 6 to generalize to unseen domains does not require training data as $\{x_i, y_i, d_i\} \sim P(X_s, Y_s, D_s)$, and neither starts with the assumptions that different populations define the source dataset. We will deal with *single-source* domain generalization.

2.3.1 State of the art: approaches and methods

Khosla et al. [74] propose to learn the biases of the datasets that define the training set, and to "undo" them, learning visual world weights that are shared across datasets. Coupling their approach with Support Vector Machines (SVM, [24]), they report better accuracy than SVMs trained without their method. Xu et al. [165] propose to learn an ensemble of classifiers and, at test time, predict to which classifier's distribution the sample is closer. Muandet et al. [107] introduce the Domain-Invariant Component Analysis (DICA), an optimization procedure that relies on kernels, aimed at learning an "invariant transformation" by minimizing the differences between the different domains, but maintaining the efficiency in solving the desired task. Ghifary et al. [41] face the domain generalization problem by using autoencoders [59]. Inspired by denoising autoencoers [157], they propose to learn features that are robust to domain translations by training autoencoders to learn the transformation of one image from its original domain into multiple different domains.

Li et al. [85] show that deep neural networks [79] outperform previous approaches for the domain generalization problem, and propose to solve it via an end-to-end solution. Mancini et al. [96, 97] propose to use an ensemble of classifiers, each trained on a different source domain, and define solutions to fuse the output of the different ones when a test sample is processed. Li et al. [86] introduce the possibility of using meta-learning to better generalize to unseen domains, proposing a "domain-agnostic" procedure where domain-shifts are simulated during the training phase. Shankar et al. [134] draw inspiration from the literature on adversarial training, introducing a method where training samples are perturbed in a way that makes them less biased towards the original training distributions.

2.4 Generative Adversarial Networks

Chapter 4 and Chapter 5 detail methods that rely on GANs [45]. In this section, we cover the basics for a general understanding of the subject.

GANs are generative models, thus aim at learning a data distribution, allowing to generate new samples from it. The original formulation by Goodfellow et al. [45] is defined by the following minimax game between a network D (discriminator) and a network G (generator)

$$\min_{\theta_D} \max_{\theta_G} \mathcal{L}_{GAN} = \mathbb{E}_{x \sim p_x}[-\log D(x; \theta_D)] + \mathbb{E}_{z \sim p_z}[-\log(1 - D(G(z; \theta_G); \theta_D))] \tag{2.1}$$

that can be also approached as the alternation between two minimization problems, defined as

$$\min_{\theta_D} \mathcal{L}_D = \mathbb{E}_{x \sim p_x}[-\log D(x; \theta_D)] + \mathbb{E}_{z \sim p_z}[-\log(1 - D(G(z; \theta_G); \theta_D))] \tag{2.2}$$

$$\min_{\theta_G} \mathcal{L}_G = \mathbb{E}_{z \sim p_z}[-\log D(G(z; \theta_G); \theta_D))] \tag{2.3}$$

Intuitively, solving the optimization problem 2.2 makes D assign label 1 to samples from the real distribution and label 0 to samples generated by G; solving the optimization problem 2.3 makes G generate samples that D would classify with label 1. With a trained G, one can generate new samples by feeding it with noise samples $z \sim p_z$.

Several alternatives to the original GAN formulation [45] have been proposed. Two examples are substituting the cross-entropy loss with the least-squares loss [99] or with the Hinge loss [103]. Also more elaborated alternatives

have been introduced [3, 77, 10]. As of today, the superiority of one objective function over the others is not fully clear [95], and the main advancements on GAN research have been related to architectural choices [115] and different training procedures [170, 103, 73, 1].

Chapter 3

Curriculum Dropout

3.1 Context

We had introduced in Chapter 1 and Chapter 2 the problem of overfitting in deep neural networks and the Dropout [141] method as a possible solution. In this Chapter, the *Curriculum Dropout* method is introduced. The following section is aimed at recapping the key concepts of dropout training and introducing and outlining the rest of the Chapter.

The seminal work by Hinton et al. [60] argues that overfitting occurs as the result of excessive co-adaptation of feature detectors which manage to perfectly explain the training data. This leads to overcomplicated models which unsatisfactory fit unseen testing data points. To address this issue, the Dropout algorithm was proposed and investigated in the seminal works [60, 142].

The point of departure of our work is the intuition that the excessive co-adaptation of feature detectors, which leads to overfitting, are very unlikely to occur in the early epochs of training. Thus, Dropout seems unnecessary at the beginning of training. Inspired by these considerations, in this work

Figure 3.1. From left to right, during training (red arrows), our curriculum dropout gradually increases the amount of Bernoulli multiplicative noise, generating multiple partitions (orange boxes) within the ***dataset*** (yellow frame) and the ***feature representation*** layers (not shown here). Differently, the original dropout [60, 142] (blue arrow) mainly focuses on the hardest partition only, complicating the learning from the beginning and potentially damaging the network classification performance.

we propose to dynamically increase the number of units that are suppressed as a function of the number of gradient updates. Specifically, we introduce a generalization of the dropout scheme consisting of a temporal scheduling - a *curriculum* - for the expected number of suppressed units. By adapting in time the parameter of the Bernoulli distribution used for sampling, we smoothly increase the suppression rate as training evolves, thereby improving the generalization of the model (see Figure 3.1).

In summary, the main contributions of this Chapter are the following.

1. We address the problem of overfitting in deep neural networks by proposing a novel regularization strategy termed Curriculum Dropout, which dynamically increases the expected number of suppressed units in order to improve the generalization ability of the model.

2. We draw connections between the original dropout framework [60, 142] with regularization theory [34] and curriculum learning [9]. This provides an improved justification of (Curriculum) Dropout training, relating it to existing machine learning methods.

3. We complement our analysis with a broad experimental validation using standard image classification benchmarks, where we compare our method with the original one [60, 142] and with the anti-Curriculum [121] paradigm. As the results certify, the proposed method generally achieves a superior classification performance.

The remaining of the Chapter is outlined as follows. The connections between this work and relevant related methods are summarized in Section 3.2. Curriculum Dropout is presented in Section 3.3 and Section 3.4, providing principled connections with Curriculum Learning [9]. The experimental evaluation is carried out in Section 3.5. Conclusions and future work are presented in Section 3.6.

3.2 Connections with other methods

In Section 2.1.1 we had discussed different ways to approach Dropout training, both from the applicative and the theoretical point of view.

Many works have proposed variations of the original strategy [69, 121, 164, 163, 7, 87]. However, it is still unclear which variation improves the most with respect to the original dropout formulation [60, 142]. In many works (such as [121]) there is no real theoretical justification of the proposed approach other than favorable empirical results. Therefore, providing a sound justification still remains an open challenge. In addition, the lack of publicly available implementations (*e.g.*, [87]) make fair comparisons problematic.

A few papers do not go beyond a bare experimental evaluation of the proposed dropout variation [87, 7, 164, 69, 121], omitting to justify the soundness of their approach. Conversely, while some works are much more formal than ours [163, 161, 168], all of them rely on approximations to carry

out their analysis which is biased towards shallow models (logistic [161] or linear regression [163, 161] and matrix factorization [168]). While some of the discussed methods can be applied in tandem, there is still a lack of understanding about which one is superior—this is also due to the lack of publicly released code (as happens in [87]). Rennie et al. [121] is the most similar to our work.

Differently from the aforementioned works, in our work we both empirically prove the experimental effectiveness of our idea and provide several natural justifications to corroborate the proposed dropout generalization for deep neural networks.

3.3 A time scheduling for the dropout rate

Deep Neural Networks display co-adaptations between units in terms of concurrent activations of highly organized clusters of neurons. During training, the latter specialize themselves in detecting certain details of the image to be classified, as shown by Zeiler and Fergus [167]. They visualize the high sensitivity of certain filters in different layers in detecting dogs, people's faces, wheels and more general ordered geometrical patterns [167, Fig. 2]. Moreover, such co-adaptations are highly generalizable across different datasets as proved by Torralba's work [169]. Indeed, the filter responses provided in the AlexNet within *conv1*, *pool2/5* and *fc7* layers are very similar [169, Fig. 5], despite the images used for the training are very different: objects from ImageNet versus scenes from Places datasets.

These arguments support the existence of some *positive* co-adaptations between neurons in the network. Nevertheless, as soon as the training keeps going, some co-adaptations can also be *negative* if excessively specific of the

training images exploited for updating the gradients. Consequently, exaggerated co-adaptations between neurons weaken the network generalization capability, ultimately resulting in overfitting. To prevent it, Dropout [60, 142] precisely contrasts those negative co-adaptations by randomly suppressing units, which in turn reflects into better generalization capabilities [60, 142].

Network training is a dynamic process. Despite the previous interpretation is rigorous, the original Dropout algorithm cannot precisely accommodate for it. Indeed, the suppression of a neuron in a given layer is modeled by a Bernoulli(θ) random variable[1], $0 < \theta \leq 1$. Employing such distribution is very natural, since it statistically models binary activation/inhibition processes. In spite of that, it seems suboptimal that θ should be *fixed* during the whole training stage. With this operative choice, [60, 142] is actually treating the negative co-adaptations phenomena as uniformly distributed during the whole training time.

Differently, our intuition is that, *at the beginning of the training, if any co-adaptation between units is displayed, this should be preserved* as positively representing the self-organization of the network parameters towards their optimal configuration.

We can understand this by considering the random initialization of the network's weights. They are statistically independent and actually not co-adapted at all. Also, it is quite unnatural for a neural network with random weights to overfit the data. On the other hand, the risk of excessive co-adaptations increases as the training proceeds since the loss minimization can achieve a small objective value by overcomplicating the hierarchical representation learnt from data. This implies that *overfitting caused by*

[1]To avoid confusion in our notation, please note that θ is the equivalent of p in [60, 142, 161], i.e the probability of *retaining* a neuron.

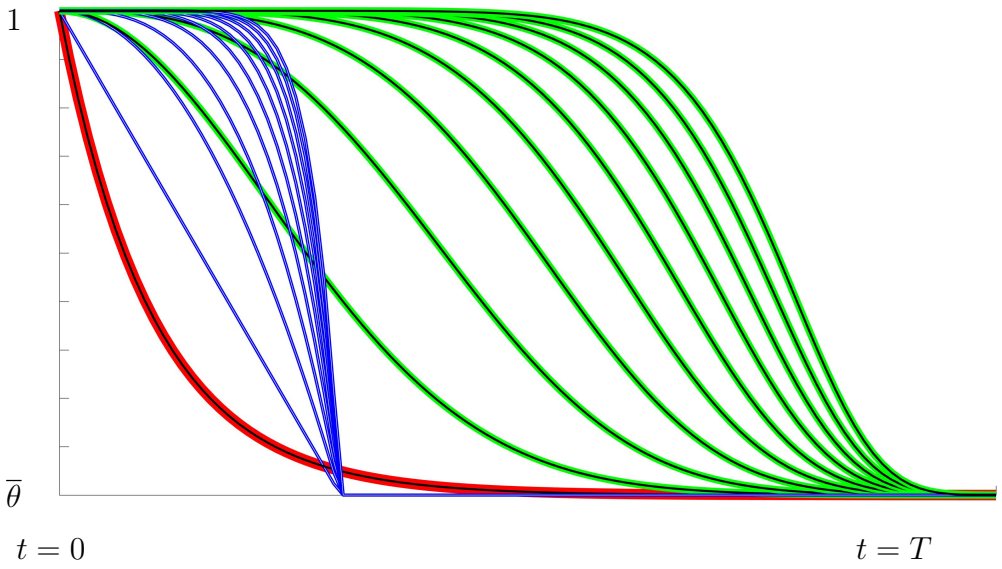


Figure 3.2. Curriculum functions. Eq. (3.1) (red), polynomial (blue) and exponential (green).

excessive co-adaptations appears only after a while.

Since a fixed parameter θ is not able to handle increasing levels of negative co-adaptations, in this work, we tackle this issue by proposing a temporal dependent $\theta(t)$ parameter. Here, t denotes the training time, measured in gradient updates $t \in \{0, 1, 2, \dots\}$. Since $\theta(t)$ models the probability for a given neuron to be retained, $D \cdot \theta(t)$ will count the average number of units which remain active over the total number D in a given layer. Intuitively, such quantity must be higher for the first gradient updates, then starting decreasing as soon as the training gears. In the late stages of training, such decrease should be stopped. We thus constrain $\theta(t)$ to be $\theta(t) \geq \overline{\theta}$ for any t, where $\overline{\theta}$ is a limit value, to be taken as $0.5 \leq \overline{\theta} \leq 0.9$ as prescribed by the original dropout scheme [142, §A.4] (the higher the layer hierarchy, the lower the retain probability).

Inspired by the previous considerations, we propose the following definition for a ***curriculum function*** $\theta(t)$ aimed at improving dropout training (as

it will become clear in section 3.4, from now on we will often use the terms *curriculum* and *scheduling* interchangeably).

Definition 1. *Any function* $t \mapsto \theta(t)$ *such that* $\theta(0) = 1$ *and* $\lim_{t\to\infty} \theta(t) \searrow \overline{\theta}$ *is said to be a curriculum function to generalize the original dropout [60, 142] formulation with retain probability* $\overline{\theta}$*.*

Starting from the initial condition $\theta(0) = 1$ where no unit suppression is performed, dropout is gradually introduced in a way that $\theta(t) \geq \overline{\theta}$ for any t. Eventually (*i.e.*when t is big enough), the convergence $\theta(t) \to \overline{\theta}$ models the fact that we retrieve the original formulation of [60, 142] as a particular case of our curriculum.

Among the functions as in Def. 1, in our work we fix

$$\theta_{\text{curriculum}}(t) = (1 - \overline{\theta}) \exp(-\gamma t) + \overline{\theta},\ \gamma > 0 \tag{3.1}$$

By considering Figure 3.2, we can provide intuitive and straightforward motivations regarding our choice.

The blue curves in Fig. 3.2 are polynomials of increasing degree $\delta = \{1, \ldots, 10\}$ (left to right). Despite fulfilling the initial constraint $\theta(0) = 1$, they have to be manually thresholded to impose $\theta(t) \to \overline{\theta}$ when $t \to \infty$. This introduces two more (undesired) parameters (δ and the threshold) with respect to [60, 142], where the only quantity to be selected is $\overline{\theta}$.

The very same argument discourages the replacement of the variable t by t^α in (3.1), (green curves in Fig. 3.2, $\alpha = \{2, \ldots, 10\}$, left to right). Moreover, by evaluating the area under the curve, we can intuitively measure how aggressively the green curves behave while delaying the dropping out scheme they eventually converge to (as $\theta(t) \to \overline{\theta}$). Precisely, that convergence is faster while moving to the green curves more on the left, being the fastest one achieved by our scheduling function (3.1) (red curve, Fig. 3.2).

One could still argue that the parameter $\gamma > 0$ is annoying since it requires cross validation. This is not necessary: in fact, γ can actually be *fixed* according to the following heuristics. Despite Def. 1 considers the limit of $\theta(t)$ for $t \to \infty$, such condition has to be operatively replaced by $t \approx T$, being T the total number of gradient updates needed for optimization. It is thus totally reasonable to assume that the order of magnitude of T is a priori known and fixed to be some power of 10 such as $10^4, 10^5$. Therefore, for a curriculum function as in Def. 1, we are interested in furthermore imposing $\theta(t) \approx \overline{\theta}$ when $t \approx T$. Actually, a rule of thumb such as

$$\gamma = 10/T \tag{3.2}$$

implies $|\theta_{\text{curriculum}}(T) - \overline{\theta}| < 10^{-4}$ and was used for all the experiments[2] in Section 3.5. Additionally, from Figure 3.2, we can grab some intuitions about the fact that the asymptotic convergence to $\overline{\theta}$ is indeed realized for a quite consistent part of the training and well before $t \approx T$. This means that during a big portion of the training, we are actually dropping out neurons as prescribed in [60, 142], addressing the overfitting issue. In addition to these arguments, we will provide complementary insights on our scheduled implementation for dropout training.

Smarter initialization for the network weights. The problem of optimizing deep neural networks is non-convex due to the non-linearities (ReLUs) and max-pooling operations. In spite of that, a few theoretical papers have investigated this issue under a sound mathematical perspective. For instance, under mild assumptions, Haeffele and Vidal [52] derive sufficient conditions to ensure that a local minimum is also a global one to guarantee that the

[2]See Appendix A, where we prove that our approach is extremely robust with respect to different γ values.

former can be found when starting from *any* initialization. The same theory presented in [52] cannot be straightforwardly applied to the dropout case due to the pure deterministic framework of the theoretical analysis that is carried out. Therefore, it is still an open question whether all initializations are equivalent for the sake of a dropout training and, if not, which ones are preferable. Far from providing any theoretical insight in this flavor, we posit that Curriculum Dropout can be interpreted as a smarter initialization. Indeed, we implement a soft transition between a classical dropout-free training of a network versus the dropout one [60, 142]. Under this perspective, our curriculum seems equivalent to performing dropout training of a network whose weights have already been slightly optimized, evidently resulting in a better initialization for them.

As a naive approach, one can think to perform regular training for a certain amount of gradient updates and then apply dropout during the remaining ones. We call that *Switch-Curriculum*. This actually induces a discontinuity in the objective value which can damage the performance with respect to the smooth transition performed by our curriculum defined by Eq. (3.1) (see Fig. 3.4).

Curriculum Dropout as adaptive regularization. Several connections [161, 162, 142, 168] have been established between Dropout and model training with noise addition [12, 122, 168]. The common trend discovered is that when an unregularized loss function is optimized to fit artificially corrupted data, this is actually *equivalent* to minimize the same loss augmented by a data dependent penalizing term. In both [161, Table 2] for linear/logistic regression and [142, §9.1] for least squares, it is proved that Dropout induces a regularizer which is scaled by $\theta(1-\theta)$. In Appendix A we extend such result for a deep

neural network, also allowing for a time-dependent $\theta(t)$.

When $\theta = \overline{\theta}$, the impact of the regularization is *fixed*, therefore rising potential over- and under-fitting issues [34]. But, for $\theta = \theta_{\text{curriculum}}(t)$, when t is small, the regularizer is set to zero ($\theta_{\text{curriculum}}(0) = 1$) and we *do not* perform any regularization at all. Indeed, the latter is simply not necessary: the network weights still have values which are close to their random and statistically independent initialization. Hence, overfitting is unlikely to occur at early training steps. Differently, we should expect it to occur as soon as training proceeds: by using (3.1), the regularizer is now weighted by

$$\theta_{\text{curriculum}}(t)(1 - \theta_{\text{curriculum}}(t)), \tag{3.3}$$

which is an increasing function of t. Therefore, the more the gradient updates t, the heavier the effect of the regularization. This is the reason why overfitting is better tackled by the proposed curriculum. Despite the overall idea of an adaptive selection of parameters is not novel for either regularization theory [54, 25, 18, 140, 21] or tuning of network hyper-parameters (e.g. learning rate, [20]), to the best of our knowledge, this is the first time that this concept of time-adaptive regularization is applied to deep neural networks.

Compendium. Let us conclude with some general comments. We posit that there is no overfitting at the beginning of the network training. Therefore, differently from previous work [60, 142], we define a scheduled retain probability $\theta(t)$ which gradually drops neurons out. Among other plausible curriculum functions as in Def. 1, the proposed choice (3.1) introduces no additional parameter to be tuned and implicitly provides a smarter weight initialization for dropout training.

The superiority of (3.1) also relates to (i) the smoothly increasingly amount of units suppressed and (ii) the soft adaptive regularization performed to contrast overfitting.

Throughout these interpretations, we can retrieve a common idea of smoothly changing difficulty of the training which is applied to the network. This idea can be better understood by finding the connections with Curriculum Learning [9], as we explain in the next section.

3.4 Curriculum Learning, Curriculum Dropout

For the sake of clarity, let us remind the concept of Curriculum Learning [9]. Within a classical machine learning algorithm, all training examples are presented to the model in an unordered manner, frequently applying a random shuffling. Actually, this is very different from what happens for the human training process, that is education. Indeed, the latter is highly structured so that the level of difficulty of the concepts to learn is proportional to the *age* of the people, managing easier knowledge when babies and harder when adults. This "start small" paradigm will likely guide the learning process [9].

Following the same intuition, [9] proposes to subdivide the training examples based on their difficulty. Then, the learning is configured so that easier examples come first, eventually complicating them and processing the hardest ones at the end of the training. This concept is formalized by introducing a learning time $\lambda \in [0, 1]$, so that training begins at $\lambda = 0$ and ends at $\lambda = 1$. At time λ, $Q_\lambda(z)$ denotes the distribution which a training example z is drawn from. The notion of curriculum learning is formalized requiring that Q_λ ensures a sampling of examples z which are easier than the ones sampled from $Q_{\lambda+\varepsilon}$, $\varepsilon > 0$. Mathematically, this is formalized by assuming

$$Q_\lambda(z) \propto W_\lambda(z)P(z). \tag{3.4}$$

In (A.20), $P(z)$ is the target training distribution, accounting for all examples,

both easy and hard ones. The sampling from P is corrected by the factor $0 \leq W_\lambda(z) \leq 1$ for any λ and z. The interpretation for $W_\lambda(z)$ is the measure of the difficulty of the training example z. The maximal complexity for a training example is fixed to 1 and reached at the end of the training, *i.e.*$W_1(z) = 1$, *i.e.*$Q_1(z) = P(z)$. The relationship

$$W_\lambda(z) \leq W_{\lambda+\varepsilon}(z) \tag{3.5}$$

represents the increased complexity of training examples from instant λ to $\lambda + \varepsilon$. Moreover, the weights $W_\lambda(z)$ must be chosen in such a way that

$$H(Q_\lambda) < H(Q_{\lambda+\varepsilon}), \tag{3.6}$$

where Shannon's entropy $H(Q_\lambda)$ models the fact that the quantity of information exploited by the model during training increases with respect to λ.

In order to prove that our scheduled dropout fulfills this definition, for simplicity, we will consider it as applied to the input layer only. This is not restrictive since the same considerations apply to any intermediate layer, by considering that each layer trains the feature representation used as input by the subsequent one.

As the images exploited for training, consider the partitions in the dataset including all the (original) clean data and all the possible ways of corrupting them through the Bernoulli multiplicative noise (see Fig. 3.1). Let π denote the probability of sampling an uncorrupted d-dimensional image within an image dataset (nothing more than a uniform distribution over the available training examples). Let us fix the gradient update t. The case of sampling a dropped-out z is equivalent to sampling the corresponding uncorrupted image z_0 from π and then overlapping it with a binary mask b (of size d), where each entry of b is zero with probability $1 - \theta(t)$. By mapping b to the number

i of its zeros,

$$\mathbb{P}[z] = \mathbb{P}[z_0, i] = \binom{d}{i}(1-\theta(t))^i\theta(t)^{d-i} \cdot \pi(z_0). \tag{3.7}$$

Indeed, $(1-\theta(t))^i\theta(t)^{d-i}$ is the probability of sampling *one* binary mask b with i zeros and $\binom{d}{i}$ accounts for all the possible combinations. Re-parameterizing the training time $t = \lambda T$, we get

$$Q_\lambda(z) = \binom{d}{i}(1-\theta(\lambda T))^i\theta(\lambda T)^{d-i} \cdot \pi(z_0). \tag{3.8}$$

By defining $P(z) = Q_1(z)$ and

$$W_\lambda(z) = \frac{1}{P(z)}\binom{d}{i}(1-\theta(\lambda T))^i\theta(\lambda T)^{d-i} \cdot \pi(z_0), \tag{3.9}$$

we can easily prove (refer to the Appendix A for the complete proof) that the definition in [9] is fulfilled by the choice (3.8) for curriculum learning distribution $Q_\lambda(z)$.

To conclude, we give an additional interpretation to Curriculum Dropout. At $\lambda = 0$, $\theta(0) = 1$ and no entry of z_0 is set to zero. This clearly corresponds to the easiest available example, since the learning starts at $t = 0$ by considering all possible available visual information. When θ start decreasing to $\theta(\lambda T) \approx 0.99$, only 1% of z_0 is suppressed (on average) and still almost all the information of the original dataset $\mathcal{Z}_0$ is available for training the network. But, as λ grows, $\theta(\lambda T)$ decreases and a bigger number of entries are set to zero. This complicates the task, requiring an improved effort from the model to capitalize from the reduced uncorrupted information which is available at that stage of the training process.

After all, this connection between Dropout and Curriculum Learning was possible thanks to our generalization through Def. 1. Consequently, the original Dropout [60, 142] can be interpreted as considering the single specific value $\overline{\lambda}$ such that $\theta(\overline{\lambda}T) = \overline{\theta}$, being $\overline{\theta}$ the constant retain probability on

[60, 142]. This means that, as previously found for the adaptive regularization (see Section 3.3), the level of difficulty $W_{\overline{\lambda}}(z)$ of the training examples z is fixed in the original Dropout. This encounters the concrete risk of either oversimplifying or overcomplicating the learning, with detrimental effects on the model's generalization capability. Hence, the proposed method allows to setup a progressive curriculum $Q_\lambda(z)$, complicating the examples z in a smooth and adaptive manner, as opposed to [60, 142], where such complication is fixed to equal the maximal one from the very beginning (Fig. 3.1).

To conclude, let us note that the aforementioned work [121] proposes a linear *increase* of the retain probability. According to equations (A.20-A.21) this implements what Bengio et al. [9] define "anti-curriculum". This is shown to perform slightly better or worse than the no-curriculum strategy [9] and always worse than any curriculum implementation. Our experiments confirm this finding.

3.5 Experiments

In this Section, we report our experiments. We applied Curriculum Dropout to neural networks for image classification problems on different datasets, using Convolutional Neural Networks (CNN, [80]) and Multi-Layer Perceptrons (MLPs). In particular, we used two different CNN architectures: LeNet [81] and a deeper one (conv-maxpool-conv-maxpool-conv-maxpool-fc-fc-softmax), further referred to as CNN-1 and CNN-2, respectively. In the following, we detail the datasets used and the network architectures adopted in each case. Code is available at `https://github.com/pmorerio/curriculum-dropout`

MNIST [82] - A dataset of grayscale images of handwritten digits (from 0 to 9), of resolution 28×28. Training and test sets contain $60,000$ and

10,000 images, respectively. For this dataset, we used a three-layer MLP, with 2,000 units in each hidden layer, and CNN-1.

Double MNIST - This is a static version of [143], generated by superimposing two random images of two digits (either distinct or equal), in order to generate 64 × 64 images. The total amount of images are 70.000, with 55 total classes (10 unique digits classes + $\binom{10}{2} = 45$ unsorted couples of digits) . Training and test sets contain 60.000 and 10.000 images, respectively. Training set's images were generated using MNIST training images, and test set's images were generated using MNIST test images. We used CNN-2.

SVHN [111] - Real world RGB images of street view house numbering. We used the cropped 32 × 32 images representing a single digit (from 0 to 9). We exploited a subset of the dataset, consisting in 60,000 images for training and 1,000 images for testing, randomly selected. We used CNN-2 also in this case.

CIFAR-10 and **CIFAR-100** [78] - These datasets collect 32 × 32 tiny RGB natural images, reporting 6,000 and 600 elements per each of the 10 or 100 classes, respectively. In both datasets, training and test sets contain 50,000 and 10,000 images, respectively. We used CNN-1 for both datasets.

Caltech-101 [35] - 300 × 200 resolution RGB images of 101 classes. For each of them, a variable size of instances is available: from 30 to 800. To have a balanced dataset, we used 20 and 10 images per class for training and testing, respectively. Images were reshaped to 128 × 128 pixels. We used CNN-2 again here.

Caltech-256 [50] - 31000 RGB images for 256 total classes. For each class, we used 50 and 20 images for training and testing, respectively. Images were reshaped to 128 × 128 pixels. We used CNN-2.

For training CNN-1, CNN-2 and MLP, we exploited a cross-entropy cost

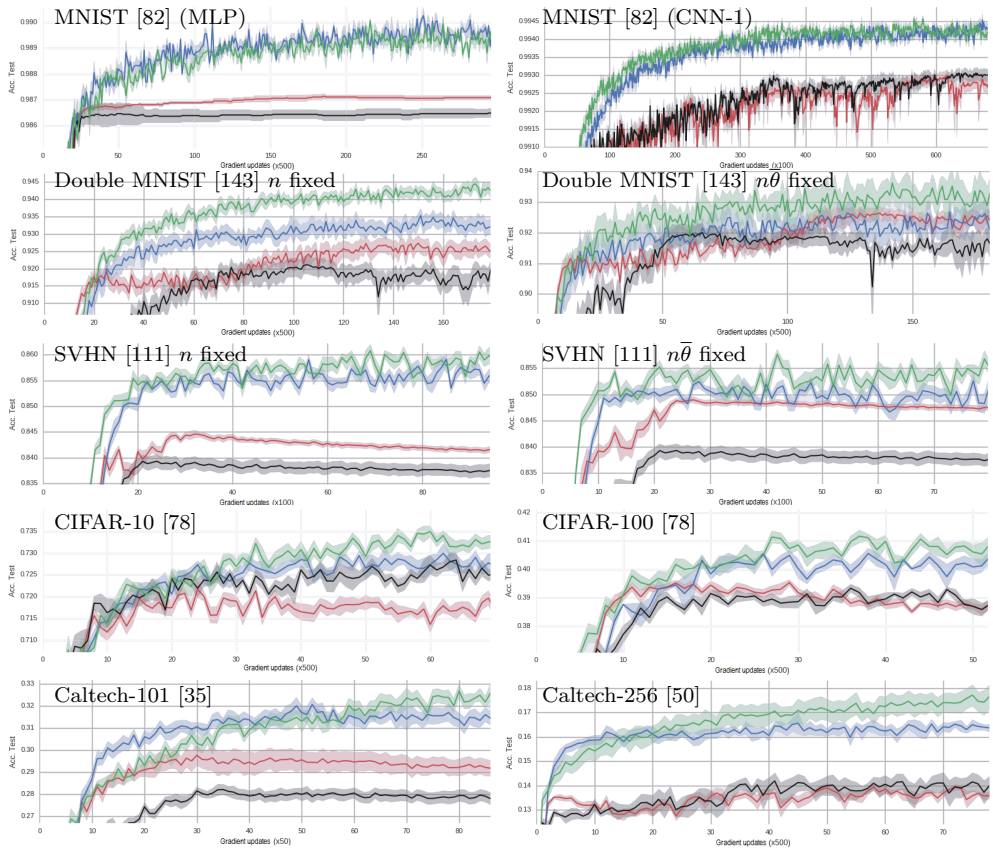


Figure 3.3. Curriculum Dropout (green) compared with regular Dropout [60, 142] (blue), anti-Curriculum (red) and a regular training of a network with no units suppression (black). For all cases, we plot mean test accuracy (averaged over 10 different re-trainings) as a function of gradient updates. Shadows represent standard deviation errors. Best viewed in colors.

function with Adam optimizer [75] and a momentum term of 0.95, as suggested in [142]. We used mini-batches of 128 images and fixed the learning rate to be 10^{-4}. Please refer to the Appendix A for additional details regarding the architectures and the hyperparameters.

We applied curriculum dropout using the function (3.1) where γ is picked using the heuristics (3.2) and $\overline{\theta}$ is fixed as follows. For both CNN-1 and CNN-2, the retain probability for the input layer was set to $\overline{\theta}_{\text{input}} = 0.9$, selecting $\overline{\theta}_{\text{conv}} = 0.75$ and $\overline{\theta}_{\text{fc}} = 0.5$ for convolutional and fully connected layers, respectively. For the MLP, $\overline{\theta}_{\text{input}} = 0.8$ and $\overline{\theta}_{\text{hidden}} = 0.5$. In all cases,

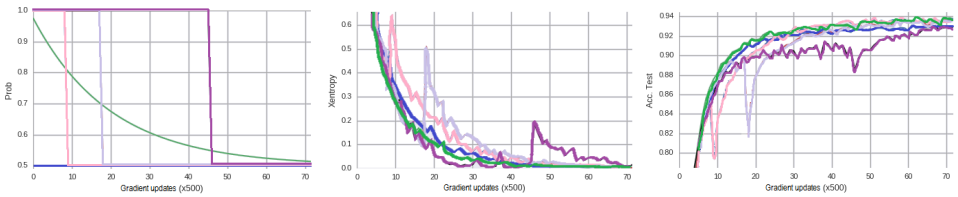


Figure 3.4. Switch-Curriculum. We compare the Curriculum (green) and the regular Dropout (blue) on Double-MNIST, in three cases where we switch from regular to dropout training i) at the beginning (pink) ii) in the middle (violet), iii) almost at the end (purple) of the learning. From left to right, curriculum functions, cross-entropy loss and test accuracy curves.

we adopted the recommended values [142, §A.4].

Before reporting our results, let us emphasize that our aim is to improve the standard dropout framework [60, 142], not to compete for the state-of-the art performance in image classification tasks. For this reason, we did not use engineering tricks such as data augmentation or any particular pre-processing, and neither we tried more complex, deeper network architectures.

In Fig. 3.3, we qualitatively compared Curriculum Dropout (green) versus the original Dropout [60, 142] (blue), anti-Curriculum Dropout (red) and an unregularized, *i.e.*no Dropout, training of a network (black). Since CNN-1, CNN-2 and MLP are trained from scratch, in order to ensure a more robust experimental evaluation, we have repeated the weight optimization 10 times for all the cases. Hence, in Fig. 3.3, we report the mean accuracy value curves, representing with shadows the standard deviation errors.

Additionally, we report in Table 3.1 the percentage accuracy improvements of Dropout [60, 142], anti-Curriculum Dropout [121] and Curriculum Dropout (proposed) versus a baseline network where no neuron is suppressed. To do that, we selected the average of the 10 highest mean accuracies obtained by each paradigm during each trial; then we averaged them over the 10 runs. We measure the boost in accuracy over [60, 142], accommodating the metric of Torralba and Efros [152]. Also, we reproduced for two datasets the cases

Dataset	Architecture	Configuration (n or $n\overline{\theta}$ fixed)	Classes	Unregularized network	Dropout [60, 142]	Anti-Curriculum	*Curriculum Dropout* Percent boost over Dropout [60, 142]
MNIST [82]	MLP	n	10	98.67	**+0.38**	+0.04	*+0.36* (-5.3%)
	CNN-1	n		99.25	+0.15	-0.05	***+0.18 (20.0%)***
Double MNIST	CNN-2	n	55	92.48	+1.42	+0.73	***+2.35 (65.5%)***
	CNN-2	$n\overline{\theta}$			+0.87	+0.53	***+1.11 (27.6%)***
SVHN [111]	CNN-2	n	10	84.63	+2.35	+1.17	***+2.65 (12.8%)***
	CNN-2	$n\overline{\theta}$			+1.59	+1.51	***+2.06 (29.6%)***
CIFAR-10 [78]	CNN-1	n	10	73.06	+0.22	-0.68	***+0.62 (182%)***
CIFAR-100 [78]	CNN-1	n	100	39.70	+1.01	+0.01	***+1.66 (64.4%)***
Caltech-101 [35]	CNN-2	n	101	28.56	+4.21	+1.57	***+4.72 (12.1%)***
Caltech-256 [50]	CNN-2	n	256	14.39	+2.36	-0.22	***+3.23 (36.9%)***

Table 3.1. Comparison of the proposed scheduling versus [60, 142] in terms of percentage accuracy improvement.

of fixed layer size n or fixed $n\overline{\theta}$ as in [142, §7.3]. Here the network layers' size n is preliminary increased by a factor $1/\overline{\theta}$, since on average a fraction $\overline{\theta}$ of the units is dropped out. However, we notice that those bigger architectures tend to overfit the data.

Switch-Curriculum. Figure 3.4 shows the results obtained on Double MNIST dataset by scheduling the dropout with a step function, *i.e.*no suppression is performed until a certain *switch-epoch* is reached (Section 3.3). Precisely, we switched at 10-20-50 epochs. This curriculum is similar to the one induced by the polynomial functions of Figure 3.2: in fact, both curves have a similar shape and share the drawback of a threshold to be introduced.

Yet, Switch-Curriculum shows an additional shortcoming: as highlighted by the spikes of both training and test accuracies, the sudden change in the network connections, induced by the sharp shift in the retain probabilities, makes the network lose some of the concepts learned up to that moment. While early switches are able to recover quickly to good performances, late ones are deleterious. Moreover, we were not able to find any heuristic rule for the *switch-epoch*, which would then be a parameter to be validated. This makes Switch-Curriculum a less powerful option compared to a smoothly-scheduled curriculum.

Discussion. The proposed Curriculum Dropout, implemented through the scheduling function (3.1), improves the generalization performance of the original algorithm [60, 142] in almost all cases. As the only exception, in MNIST [82] with MLP, the scheduling is just equivalent to the original dropout framework [60, 142]. Our guess is that the simpler the learning task, the less effective Curriculum Learning. After all, for a task which is relatively easy itself, there is less need for "starting easy". This is in any case done at no additional cost nor training time requirements.

As expected, anti-Curriculum was improved by a more significant gap by our scheduling. Also, sometimes, an anti-Curriculum strategy even performs worse than a non-regularized network (*e.g.*, Caltech 256 [50]). This is coherent with the findings of [9] and with our discussion in Section 3.4 concerning Annealed Dropout [121], of which anti-Curriculum represents a generalization. In addition, while neither regular nor Curriculum Dropout ever need early stopping, anti-Curriculum often does.

3.6 Conclusions and future work

We propose a scheduling for dropout training applied to deep neural networks. By softly increasing the amount of units to be suppressed layerwise, we achieve an adaptive regularization and provide a better smooth initialization for weight optimization. This allows us to implement a mathematically sound curriculum [9] and justifies the proposed generalization of [60, 142].

Through a broad experimental evaluation on 7 image classification tasks, the proposed Curriculum Dropout have proved to be more effective than both the original Dropout [60, 142] and the Annealed [121], the latter being an example of anti-Curriculum [9] and therefore achieving an inferior performance to our more disciplined approach in ease dropout training. Globally, we always outperform the original Dropout [60, 142] using various architectures, and we improve the idea of [121] by margin.

Our guess is that, as standard Dropout, our method is very general and thus applicable to domains different than the image one, considered in this Chapter. As future work, we plan to apply our scheduling to other computer vision tasks, also extending it for the case of inter-neural connection inhibitions [162] and Recurrent Neural Networks.

Chapter 4

Adversarial Feature Augmentation

4.1 Context

We had defined in Chapter 2 Generative Adversarial Networks (GANs [45]) as generative models defined by a generator (G) and a discriminator (D), with a training procedure designed as a minimax game where G is optimized to fool D, and D is optimized to correctly classify generated samples from actual training samples. Recently, this framework proved to be able to generate images with impressive accuracy [115, 103, 1], to generate videos from static frames [160], and to translate images from one style to another [148, 90, 16, 89, 170].

Furthermore, GANs have been exploited in the context of unsupervised domain adaptation. As we had discussed in Section 2.2, in unsupervised domain adaptation a source (labeled) dataset and a target (unlabeled) dataset are considered, which are separated by the so-called domain shift [152], *i.e.*, they are drawn from two different data distributions. The goal is to build

models that are able to correctly classify target samples, despite the domain shift. In this framework, adversarial training has been used (i) to learn feature extractors that map target samples in a feature space indistinguishable from the one where source samples are mapped (*e.g.*, [39, 153]), and (ii) to develop image-to-image translation algorithms aimed at converting source images into a style that resembles that of the target image domain, and vice-versa (*e.g.*, [148, 90, 16, 89]).

In this Chapter, we present a new training procedure that builds on the work by Tzeng et al. [153], which proposes to use a GAN objective to learn target features that are indistinguishable from the source ones, leading to a pair of feature extractors, one for the source and one for the target samples. We extend this approach in two directions: (a) we force domain-invariance in a *single* feature extractor trained through GANs, and (b) we perform data augmentation in the feature space (*i.e.*, *feature augmentation*), by defining a more complex minimax game. More specifically, we perform feature augmentation by devising a feature generator trained with a conditional GAN (cGAN [102]). The minimax game is here played with features instead of images, allowing to generate features conditioned to the desired classes. The cGAN generator is thus able to learn the class distribution in the feature space, and therefore to generate an arbitrary number of labeled feature vectors. Our results show that forcing domain-invariance and augmenting features are both valuable approaches in the unsupervised domain adaptation setting, leading to higher classification accuracies.

In summary, the main contributions we bring in this Chapter are:

1. Introducing for the first time the use of GANs to perform data augmentation in the feature space.

2. Proposing a new method for unsupervised domain adaptation, based on

feature augmentation and (source/target) feature domain-invariance.

3. Evaluating the proposed method on unsupervised domain adaptation benchmarks (cross-dataset digit classification and cross-modal object classification).

The remaining of the Chapter is organized as follows. Section 4.2 is dedicated to the related work. The models and the training procedure are presented in Section 4.3. In Section 4.4, the datasets used for the analysis and method's validation are described. The experiments and associated results are detailed in Section 4.5. Finally, conclusive remarks are drawn in Section 4.6.

4.2 Connections with other works

The work related to our proposed method is focused on GAN research and on modern domain adaptation techniques (*i.e.*, based on deep learning).

Generative adversarial networks. In the original formulation by Goodfellow et al. [45], a GAN model is trained through a minimax game between a generator, that maps noise vectors in the image space, and a discriminator, trained to discriminate generated images from real ones. Several other papers address ways to control what GANs generate [102, 22, 120]. In particular, cGANs [102] allow to condition on the desired classes, from which samples are generated. Other works [32, 30] propose to learn inference by playing a minimax game against features. In these works, trained models are feature extractors that map images into the feature space, not feature generators, which is our primary goal.

Performing feature augmentation through GANs is one of the original aspects of our approach. We propose a generator able to generate features

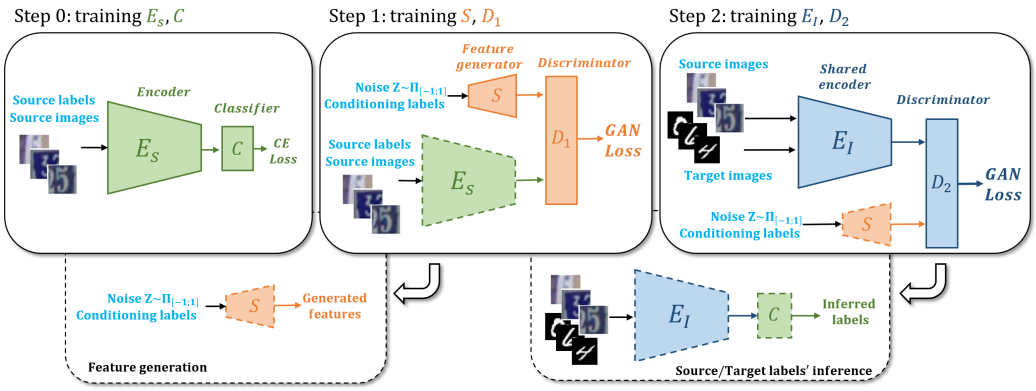


Figure 4.1. Training procedure, representing the steps described in Section 4.3.1. Solid lines indicate that the module is being trained, dashed lines indicate that the module is already trained (from previous steps). All modules are neural networks, whose architectures are detailed in Section A.2.2. Smaller, dashed panels in the bottom indicate how to generate features (*left*) and how to infer source or target labels (*right*).

from noise vectors and label codes, via a cGAN [102] framework, playing a minimax game with features extracted from a pre-trained model instead of images.

Unsupervised domain adaptation. Naturally, the literature related to unsupervised domain adaptation is closely related to our work. In Section 2.2 we had discussed different approaches to face this task.

As mentioned in the previous section, the approach we are proposing here is mostly related to the methods which aim as bridging the domain gap via adversarial training [39, 40, 153]. In particular, the domain-invariant feature extractor we designed is inspired by Tzeng et al. [153], with two main differences. First, we play the minimax game against features which are *generated* by a pre-trained model, thus performing *feature augmentation*. Second, we train the feature extractor in order to make it work for both source and target samples (thus achieving *domain-invariance*), avoiding catastrophic forgetting. Both modifications lead to higher accuracies in classifying target

samples, as we will show in Section 4.5. Domain-invariance also allows to use the same feature extractor for both source and target samples, while in Tzeng et al. [153] two different encoders are required.

4.3 The DIFA method

Our goal is to train a domain-invariant feature extractor (E_I), whose training procedure is made more robust by data augmentation in the space of source features. The training procedure we designed to accomplish our intent is based on three different steps, depicted in Figure 4.1. First, we need to train a feature extractor on source data ($C \circ E_s$). This step is necessary because we need a reference feature space and a reference classifier that performs well on it. Secondly, we need to train a feature generator (S) to perform data augmentation in the source feature space. We can train it by playing a GAN minimax game against features extracted through E_S. Finally, we can train a domain-invariant feature extractor (E_I) by playing a GAN minimax game against features generated through S. This module can then be combined with the softmax layer previously trained ($C \circ E_I$) to perform inference on both source and target samples. All modules are neural networks trained by backpropagation [126]. In the following sections, we detail how each Step is performed, how new features can be generated, and how source/target labels can be inferred.

4.3.1 Training procedure

Step 0. The model $C \circ E_s$ is trained to classify source samples. E_s represents a CNN [80] feature extractor and C represents a fully connected softmax layer, with a size that depends on the problem. The optimization problem

consists in the minimization of the following cross-entropy loss (*CE Loss* in Figure 4.1):

$$\min_{\theta_{E_s},\theta_C} \ell_0 = \mathbb{E}_{x_i,y_i\sim P_s(X,Y)} H(C \circ E_s(x_i), y_i), \tag{4.1}$$

where θ_{E_s} and θ_C indicate the parameters of E_s and C, respectively, $P_s(X,Y)$ is the joint distributions of source samples and source labels, respectively, and H represents the softmax cross-entropy function.

Step 1. The model S is trained to generate feature samples that resemble the source features. Exploiting the cGAN framework, the following minimax game is defined:

$$\begin{aligned}\min_{\theta_S}\max_{\theta_{D_1}} \ell_1 = {} & \mathbb{E}_{z,y_i\sim P_z(Z),P_s(Y)} \|D_1(S(z||y_i)||y_i) - 1\|^2 \\ & + \mathbb{E}_{x_i,y_i\sim P_s(X,Y)} \||D_1(E_s(x_i)||y_i)\|^2,\end{aligned} \tag{4.2}$$

where θ_S and θ_{D_1} indicate the parameters of S and D_1, respectively, $P_z(Z)$ is the distribution[1] from which noise samples are drawn, and $\|$ denotes a concatenation operation. In this and the following steps, we relied on Least Squares GANs [99] since we observed more stability during training.

Feature generation. In order to generate an arbitrary number of new feature samples, we only need S, which takes as input the concatenation of a noise vector and a *one-hot* label code, and outputs a feature vector from the desired class:

$$f_i(z|y_i) = S(z||y_i) \tag{4.3}$$

where $z \sim P_z(Z)$ and f_i is a feature vector belonging to the class label associated with y_i (dashed box in Figure 4.1, *left*).

[1]Uniform in the range $[-1, 1]$ throughout this Chapter and the following one.

Step 2. The domain-invariant encoder E_I is trained via the following minimax game, after being initialized with weights optimized on Step 0 (note that E_S and E_I have the same architecture), a requirement to reach optimal convergence:

$$\min_{\theta_{E_I}} \max_{\theta_{D_2}} \ell_2 = \mathbb{E}_{x_i \sim P_{s \cup t}(X)} \| D_2(E_I(x_i)) - 1 \|^2 + \mathbb{E}_{z, y_i \sim P_z(Z), P(Y)} \| D_2(S(z || y_i)) \|^2, \tag{4.4}$$

where θ_{E_I} and θ_{D_2} indicate the parameters of E_I and D_2, respectively, and $P_{s \cup t}(X)$ is the distribution of both source and target samples. Since the model E_I is trained using both source and target domains, the feature extractor results domain-invariant. In particular, it maps both source and target samples in a common feature space, where features are indistinguishable from the ones generated through S. Since the latter module is trained to produce features indistinguishable from the source ones, the feature extractor E_I can be combined with the classification layer of Step 0 (C) and used for inference (as in Tzeng et al. [153]):

$$\tilde{y}_i = C \circ E_I(x_i), \tag{4.5}$$

where x_i is a generic image from the source or the target data distribution and $\tilde{y}_i$ is the inferred label (dashed box in Figure 4.1, *right*). We term our method *DIFA* (*Domain Invariance - Feature Augmentation*).

4.4 Datasets

To evaluate our approach, we used several benchmark splits of public source/target datasets adopted in domain adaptation. We recap the details of some datasets

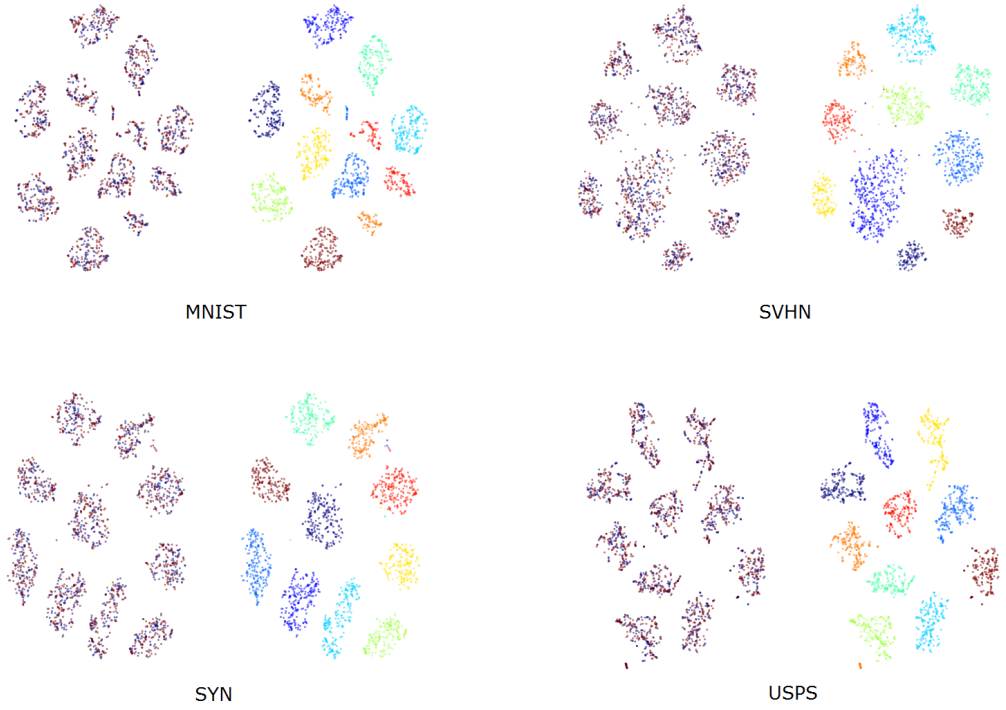


Figure 4.2. t-SNE plots of features associated with different adopted datasets (MNIST, SVHN, SYN, USPS). For each dataset, in the *left* part of the panels, red and blue dots indicate real and generated features, respectively. In the *right* part of the panels, different colors indicate different classes.

even if they were discussed also in Chapter 3.

MNIST ↔ USPS. Both datasets consist of white digits on a solid black background. We tested two different protocols: the first one (P1) consists in sampling 2,000 MNIST [82] images and 1,800 USPS [28] images. The second one (P2) consists in using the whole MNIST training set, 50,000 images, and dividing USPS in 6,562 images for training, 2.007 for testing, and 729 for validation. For P1, we tested the two directions of the split (MNIST → USPS and MNIST ← USPS). For P2, we tested only MNIST → USPS, and we avoided to use the validation set in this case, too. In both experimental protocols, we resized USPS digits to 28×28 pixels, which is the MNIST

images' size.

SVHN → MNIST. SVHN [111] is built with real images of Street View House Numbers. We used the whole training sets of both datasets, following the standard protocol for unsupervised domain adaptation (SVHN training set contains $73,257$ images), and tested on MNIST test set. We resized MNIST images to 32×32 pixels and converted SVHN to grayscale. We did not use the extra set of SVHN.

SYN DIGITS → SVHN. This split represents a synthetic-to-real domain adaptation problem, of great interest for research in computer vision since, quite often, generating labeled synthetic data requires less effort than obtaining large labeled dataset with real examples. SYN DIGITS [39] contains $500,000$ images belonging to the same SVHN classes. We tested on SVHN test set.

NYUD (RGB → D). This modality adaptation problem was proposed by Tzeng et al. [153]. The dataset is gathered by cropping out tight bounding boxes around instances of 19 object classes present in the NYUD [137] dataset. It comprises 2,186 labeled source (RGB) images and 2,401 unlabeled target (HHA-encoded [51]) depth images. Note that these are obtained from two different splits of the original dataset, to ensure that the same instance is not seen in both domains. The adaptation task is extremely challenging, due to the very different domains, the limited number of examples (especially for some classes), and the low resolution of the cropped bounding boxes.

4.5 Experiments

In this section, we evaluate our approach. First, we show that our model S is able to generate consistent and discriminant feature vectors conditioned on the desired classes. Second, we report an ablation study to figure out the benefits brought by the different steps that compose our approach. Finally, we compare our method with competing algorithms on unsupervised domain adaptation tasks. Before reporting the results, we provide the description of architectures and hyperparameters used. All models were implemented using Tensorflow, and training procedures were performed on a NVIDIA Titan X GPU. The code is available at `https://github.com/ricvolpi/adversarial-feature-augmentation`.

Architectures. The module S is built by the repetition of two blocks, each defined by a fully connected layer, a Batch Normalization layer [68], and a Dropout layer [141], followed by a fully connected layer with *tanh* activation functions. D_1 is a one-hidden-layer neural network, with a *sigmoid* hidden unit as output layer. We defined E_S and E_I following standard architectures used in unsupervised domain adaptation [39]. In particular, for SVHN $\rightarrow$ MNIST, MNIST $\rightarrow$ USPS and USPS $\rightarrow$ MNIST, we defined the network as conv-pool-conv-pool-fc-fc-softmax (with Dropout [141] on fully connected layers for MNIST $\leftrightarrow$ USPS experiments). For SYN $\rightarrow$ SVHN, conv-pool-conv-pool-conv-fc-fc-softmax. For the NYUD experiment, in order to be comparable with [153], we used a VGG-16 [138] pretrained on ImageNet [27]. The final feature dimensionality (*e.g.*, the size of the feature vector fed to the softmax layer) was set to 128 for all experiments, except for SYN $\rightarrow$ SVHN (256). D_2 is built with two or three fully connected layers (depending on the experiment) with a *sigmoid* unit on top. Note that for the NYUD experiment

we used three hidden layers, while Tzeng et al. [153] built the discriminator with two, since our method requires an additional one to reach convergence. For all our experiments, we used Adam optimizer [75] with momentum set to 0.95. *ReLU* [109] units were used throughout the architectures, except for last layers of discriminators, defined as *sigmoid* units, last layer of S, whose activation functions are *tanh*, and D_2, which was built with *Leaky ReLU* units, in agreement with the findings of Radford et al. [115]. Further detailes regarding the architectures used can be found in Appendix B.

Hyperparameters. For the digits experiments, for each training Step, we use a batch size of 64 samples. The learning rate is set to $3 \cdot 10^{-4}$ for Step 0, $1 \cdot 10^{-4}$ for Step 1 and $3 \cdot 10^{-5}$ for Step 2, in all experiments except MNIST $\leftrightarrow$ USPS, where is set to $3 \cdot 10^{-6}$. Concerning the NYUD experiment, batch size is 32 (instead of 128) due to hardware limitations. The learning rate is 10^{-4} for Step 0, 10^{-5} for Step 1 and 10^{-7} for Step 2. Note that hyperparameters were set in order to reach the convergence of the GAN [45] minimax games, no cross-validation using target labels was performed.

Table 4.1. *Second* column: number of activation patterns (APs) among the features extracted from training data. *Third* column: number of APs that S is able to generate. *Fourth* column: classification accuracy of the generated features, accordingly to given labels.

Dataset	#APs $E_S(x)$	#APs $S(z\|\|y)$	Accuracy
SVHN	$69,625$		0.974
USPS	$1,422$	$\sim \mathbf{10^6}$	0.998
MNIST	$1,910$		0.995
NYUD	19	$\sim \mathbf{10^3}$	0.998

4.5.1 Generating features

We qualitatively show with t-SNE [156] that we can generate feature vectors from the desired classes, after having trained S as described in Section 4.3.1. Figure 4.2 shows comparisons between real and generated features for different datasets. For each dataset, two identical point clouds are represented: the *bi-color* side (at the left of each panel), highlights real and generated samples (in red and blue, respectively); the *multi-color* side (at the right of each panel) highlights instead the different classes. From a qualitative point of view, real and generate features appear indistinguishable, and class structure is preserved. To quantitatively measure the quality of the features generated, we fed them to the classifier C trained with the original samples for class estimation. Table 4.1 (fourth column) shows that such features are also quantitatively reliable, and this is valid for all the datasets considered.

Feature augmentation. Finally, we are interested in evaluating the variability of the features generated through S to figure out whether (i) the model is memorizing the features from the training set, and (ii) it is realistic to assume that we are performing data augmentation in the feature space. To shed light on these two questions, we decided to perform the following empirical test: we counted the number of activation patterns (APs) that S is able to generate, and compared it with the ones intrinsically available in the original dataset. An activation pattern is defined by thresholding the output of the activation functions of the hidden state of a network. Raghu et al. [116] defined this concept for $ReLUs$ [109], where values greater than zero are set to one, the others to zero. For our purposes, we can apply the same rule even if we are using *tanh* activation functions. For example, SVHN has $73,257$ samples that - with the feature extractor we used for our experiments -

correspond to $69,625$ activation patterns. S can instead generate a number of activation patterns in the order of 10^6 (counted empirically, feeding noise to S till saturation), indistinguishable from the original ones due to the training procedure defined in Section 4.3.1. Table 4.1 reports the results associated with the other datasets considered. Interestingly, activation patterns associated with the $2,186$ source samples of NYUD are only 19: each pattern is associated with a different class. This is most likely due to overfitting: the network is already explicitly encoding classes at feature level. However, the generator S can enrich the feature set to a broad extent.

4.5.2 Ablation study

We carried out an ablation study to evaluate the benefit brought by the introduced modifications to the current way of using GAN objectives in unsupervised domain adaptation. Since the Least Squares GAN [99] framework is required to solve Step 1 and Step 2 of our method (Section 4.3.1), we re-designed the ADDA algorithm [153] in this framework as a baseline, and from this point we implemented our peculiar contributions, showing that each one favourably concurs to improve performance. We term it *LS-ADDA*, and it is defined by the following minimax game:

$$\begin{aligned}\min_{\theta_{E_t}} \max_{\theta_D} \ell = \mathbb{E}_{x_i \sim P_t(X)} \|D(E_t(x_i)) - 1\|^2 \\ + \mathbb{E}_{x_i \sim P_s(X)} \|D(E_s(x_i))\|^2,\end{aligned} \tag{4.6}$$

where E_s is the feature extractor trained on source samples (as the one pre-trained in Step 0, Figure 4.1), and E_t is the encoder for the target samples that is being trained. D is the discriminator, as those described in this work.

The second analysis stage lies in imposing domain-invariance, and this is

carried out by solving the following minimax problem:

$$\min_{\theta_{E_I}} \max_{\theta_D} \ell = \mathbb{E}_{x_i \sim P_{s \cup t}(X)} \|D(E_I(x_i)) - 1\|^2 + \mathbb{E}_{x_i \sim P_s(X)} \|D(E_s(x_i))\|^2, \tag{4.7}$$

where E_I is the *shared* encoder for the source and target samples that is being trained, and the rest of the modules are the same described above. This represents our first notable contribution, which we call *DI* (short for *DI LS-ADDA*, as this architecture introduces domain-invariance to *LS-ADDA*). Finally, the third analysis stage is constituted by our complete proposed approach, in which the minimax game also embeds the feature augmentation procedure (described in Step 2 of Section 4.3.1). For each of the three architectures proposed in this ablation study, we finally end up with an encoder that can be combined with the module C trained in Step 0 (see Figure 4.1). We tested these algorithms on the benchmark splits detailed in Section 4.4. Figure 4.3 shows the evolution of the performance of these three frameworks throughout the minimax games: *green* curves are associated with *LS-ADDA*, *orange* curves are associated with *DI*, and *blue* curves are associated with *DIFA*.

The values reported in the bottom part of the plots indicate the average and the standard deviation calculated over the final stages of training, *i.e.*, when the minimax game reaches a stability point, despite oscillations. For the splits SVHN $\rightarrow$ MNIST and SYN $\rightarrow$ SVHN, we averaged over three different runs, due to some instability in the equilibriums reached, that can be observed in Figure 4.3. The general trend is that enforcing domain-invariance (*DI*) brings a first improvement (except in the MNIST $\rightarrow$ USPS (P1) experiment), and feature augmentation (*DIFA*) adds a further increment. In NYUD, *LS-ADDA* cannot converge.

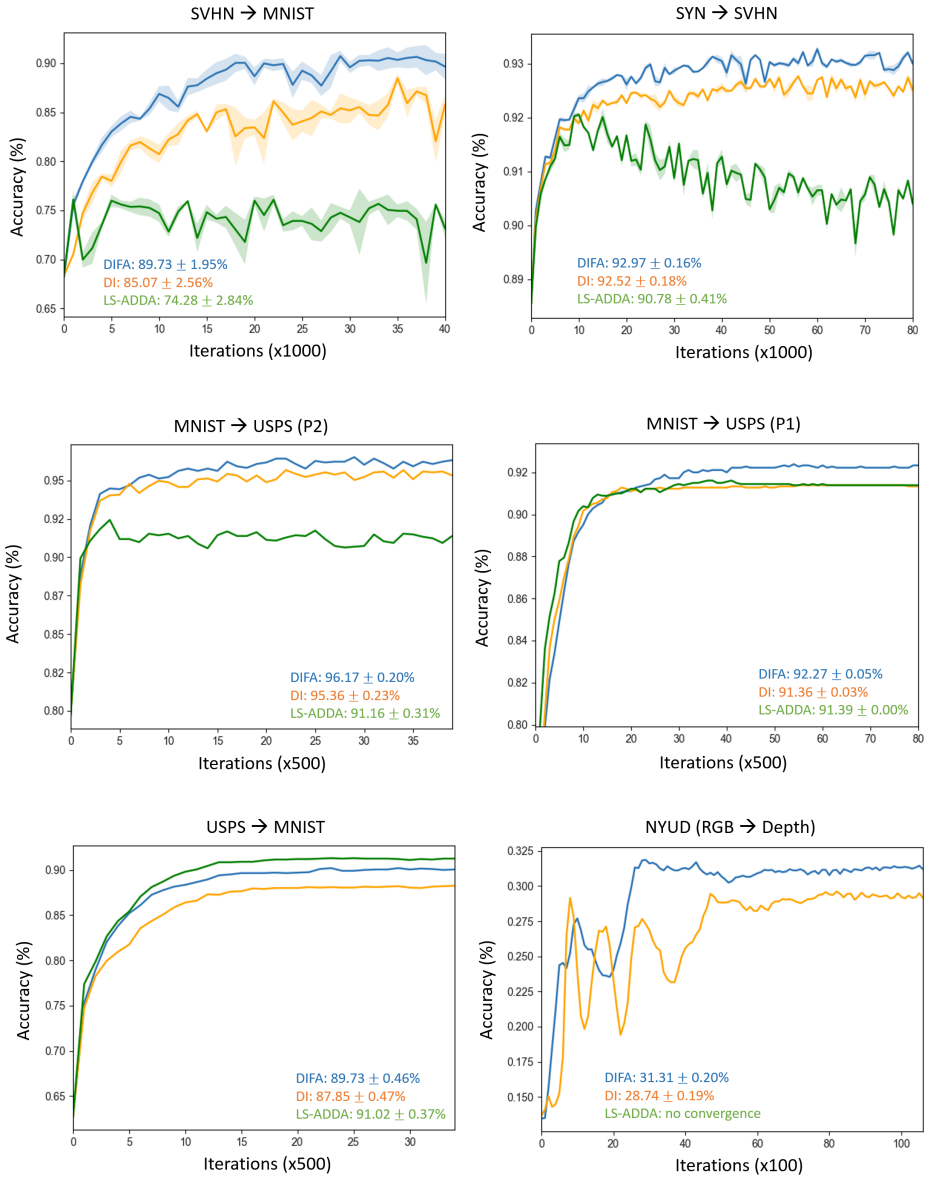


Figure 4.3. Accuracies on target samples evaluated throughout the training of the feature extractors of *LS-ADDA* (*green*), *DI* (*orange*) and DIFA (*blue*). Inference was performed by combining the feature extractor being learned with C of Step 0, Section 4.3.1. In the NYUD experiment the green curve is missing due to non-convergence of *LS-ADDA*. SVHN → MNIST and SYN → SVHN plots were obtained averaging over three different runs; confidence bands are portrayed.

The only exception is USPS → MNIST, where *LS-ADDA* is the best performing method. Note that we did not report experiments related to em-

bedding feature augmentation without domain-invariance because it performs poorly, due to high instability.

4.5.3 Comparisons with other methods

Tables 4.2 and 4.3 report our findings and results obtained by the other works in the literature[2]. In both Tables, the first row reports accuracies on target data achieved with non-adapted classifiers trained on source data, and the last row reports accuracies on target data achieved with classifiers trained on target data (*oracle*). Our main contributions lie in forcing the domain-invariance in the GAN minimax game (*DI*) and further improving it with feature augmentation (*DIFA*). A difficulty in unsupervised domain adaptation is determine the fair accuracy reached by each method, since cross-validation is not feasible (target labels should be used only to evaluate the method at the end of the training procedure). We believe that a fair way is the one we proposed in the previous section ($mean \pm std$ calculated over the last iterations), since choosing a single value would be arbitrary and unfair in stochastic training procedures (*e.g.*, see SVHN $\rightarrow$ MNIST and SYN $\rightarrow$ SVHN in Figure 4.3).

Results show that our approach based on domain-invariance and feature augmentation leads to accuracies comparable or higher to current state-of-the-art in several unsupervised domain adaptation benchmarks. Among the splits we tested, the only exception is USPS $\rightarrow$ MNIST, where ADDA [153] and our implementation of it (*LS-ADDA*) perform better - with the drawback of having two different feature extractors for source/target samples. In SVHN $\rightarrow$ MNIST, our approach gives results comparable to current state-

[2]Note that this refers to the state of the art at the time the content of this Chapter was published [158]. Updated versions of these tables can be found in the next Chapter.

Table 4.2. Comparison of our method with competing algorithms. The row *LS-ADDA* lists results obtained by our implementation of Least Squares ADDA. The row *Ours (DI)* refers to our approach in which only domain-invariance is imposed. The row *Ours (DIFA)* refers to our full proposed method, which includes feature augmentation.

	MNIST→USPS$_{P1}$	MNIST→USPS$_{P2}$	USPS→MNIST
Source	0.723	0.797	0.627
DANN [39, 40]	0.771 ± 0.018 [153]	-	0.730 ± 0.020 [153]
DDC [153]	0.791 ± 0.005	-	0.665 ± 0.033
ADDA [153]	0.894 ± 0.002	-	**0.901 ± 0.008**
PixelDA** [16]	-	0.959	-
UNIT [89]	-	0.960	-
CoGANs [90]	0.912 ± 0.008	0.957 [89]	0.891 ± 0.008
LS-ADDA	0.914 ± 0.000	0.912 ± 0.003	**0.910 ± 0.004**
Ours (DI)	0.914 ± 0.000	0.954 ± 0.002	0.879 ± 0.005
Ours (DIFA)	**0.923 ± 0.001**	0.962 ± 0.002	0.897 ± 0.005
Target	0.999	0.999	0.975

of-the-art (UNIT [89]), but it must be noted that the latter was achieved by making use of extra SVHN set (531, 131 images), making the result difficult to interpret. In MNIST → USPS (P2) we perform better or comparably to any other method that was tested on it. Also note that all those methods [16, 90, 89] rely on the generation of target images to perform adaptation, and that [16, 89] rely on additional hyperparameters - a severe drawback in unsupervised domain adaptation, where cross-validation is not applicable. In SYN → SVHN, our method is statistically comparable with the one proposed by Saito et al. [129]. In this case, it is also worth noting that the adapted feature extractor performs better than a neural network trained on SVHN (target) training set (see Table 4.3, last row). This opens a wide range of

Table 4.3. Comparison of our method with competing algorithms. The row *LS-ADDA* lists results obtained by our implementation of Least Squares ADDA. The row *Ours (DI)* refers to our approach in which only domain-invariance is imposed. The row *Ours (DIFA)* refers to our full proposed method, which includes feature augmentation. (*) DTN [148] and UNIT [89] use extra SVHN data (531,131 images). Convergence not reached is indicated as *no conv.*

	SVHN→MNIST	SYN→SVHN	NYUD (RGB → Depth)
Source	0.682	0.885	0.139
DANN [39, 40]	0.739	0.911	-
DDC [153]	0.681 ± 0.003	-	-
DSN [17]	0.827	0.912	-
ADDA [153]	0.760 ± 0.018	-	0.211
Tri [129]	0.862	**0.931**	-
DTN [148]	0.844*	-	-
UNIT [89]	**0.905***	-	-
CoGANs [90]	no conv. [153]	-	-
LS-ADDA	0.743 ± 0.028	0.908 ± 0.004	no conv.
Ours (DI)	0.851 ± 0.026	0.925 ± 0.002	0.287 ± 0.002
Ours (DIFA)	**0.897 ± 0.020**	**0.930 ± 0.002**	**0.313 ± 0.002**
Target	0.992	0.913	0.468 [153]

possibility of using synthetic data, which are much easier to obtain than labeled, real data in real-world applications. In NYUD (RGB → Depth), we perform better than ADDA [153] by a large margin. In particular, embedding both domain-invariance and feature augmentation leads to an improvement $> 10\%$. We did not include the work by Haeusser et al. [53] in the Tables because it makes use of a much more powerful feature extractor (conv-conv-pool-conv-conv-pool-conv-conv-pool-fc-softmax), which makes their method hard to compare with other works.

Table 4.4. Difference in accuracy between training and test *source* data, by classifying with $C \circ E_S$ and $C \circ E_I$. Source test data is not provided for NYUD [153]. E_I does not experience catastrophic forgetting and generalizes well on unseen source data (test).

Dataset	$E_S \rightarrow E_I(training)$	$E_S \rightarrow E_I(test)$
USPS	$0.975 \rightarrow 0.973$	$0.980 \rightarrow 0.979$
MNIST (P1)	$1.000 \rightarrow 0.997$	$0.960 \rightarrow 0.961$
MNIST (P2)	$0.997 \rightarrow 0.986$	$0.992 \rightarrow 0.984$
SVHN	$0.982 \rightarrow 0.883$	$0.905 \rightarrow 0.856$
SYN	$0.998 \rightarrow 0.996$	$0.995 \rightarrow 0.994$
NYUD	$1.000 \rightarrow 1.000$	*test set n.a.*

4.5.4 Domain invariance

Finally, Table 4.4 shows the difference of performance on classifying source samples using $C \circ E_s$ or $C \circ E_I$. As it can be observed, the encoder E_I (trained following Step 2) works well on source samples, too. This allows to use the same encoder for both target and source data, a very useful feature in an application setting where we might not know the source of the data. The worst results on source samples, achieved on SVHN dataset, are most likely due to the large difference between the source and the target domains.

4.6 Conclusion, limitations and future work

We propose two techniques to improve the current usage of GAN objectives in the unsupervised domain adaptation framework. First, we induce domain-invariance through a straightforward extension of the original algorithm. Second, we propose to perform data augmentation in the feature space through GANs [45], a novel application. An exhaustive evaluation is carried

out on standard domain adaptation benchmarks, and results confirm that both approaches lead to higher accuracy on target data. Also, we show that the obtained feature extractors can be used on source data, too.

Results show that our approach is comparable or superior to current state-of-the-art methods, with the exception of a single benchmark. In particular, we perform better than recent, more complex methods that rely on generating target images to tackle unsupervised domain adaptation tasks.

The main limit of the domain-invariant feature extractor we designed is the same that can be detected in the works by Tzeng et al. [153] and by Ganin and Lempitsky [39]. Practically, all these approaches encourage source and target features to be indistinguishable, but this does not guarantee that target samples will be mapped in the correct regions of the feature space. In our case and in ADDA's one, this strongly depends on the feature extractor trained on source samples: if the representation is far from being good, the results will be sub-optimal.

For future work, we plan to test our approach on more complex unsupervised domain adaptation problems, as well as investigate if feature augmentation can be applied to different frameworks, *e.g.*, the contexts where traditional data augmentation proved to be successful.

Chapter 5

The Bright Side of Mode Collapse

5.1 Context

The capability of Generative Adversarial Networks (GANs, [45]) to actually learn data distributions is an open debate [4, 5]. From an empirical point of view, a proven fact is that GANs tend to model only some areas of the data distribution, ignoring others [47]. This flaw is commonly referred to as "mode collapse", and it results in models generating samples with low variability.

While mode collapse is naturally considered a downside of GANs and different ways to overcome the issue have been proposed (*e.g.*, [101]), in this Chapter we explore a context where this peculiarity carries several benefits. We study the behavior of the conditional GAN model (cGAN, [102]), that allows to generate samples conditioning on the desired classes, when training data is polluted with label noise (*i.e.*, an unknown amount of training samples is provided with the wrong label). We observe that, when label pollution is reasonably below some level, cGANs are robust against this kind of noise.

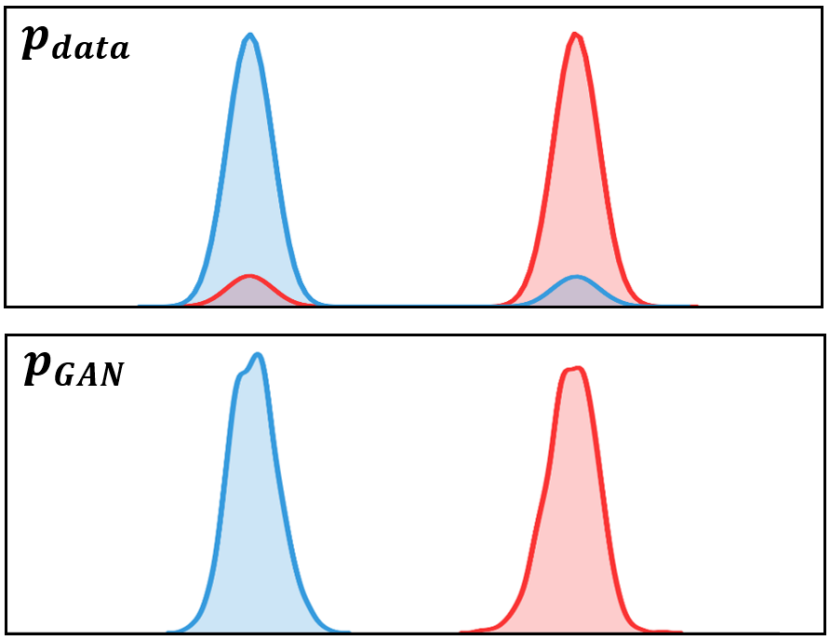


Figure 5.1. Each color is related to a different label. In the real distribution P_{data}, a set of samples is provided with the wrong labeling. Due to conditional mode collapse, the distribution learned via conditional GANs P_{GAN} is resistant to such label pollution.

This is the effect of GANs being prone to collapsing into regions of the data space with higher probability. Under the presence of label noise, the collapsing behavior results in cGANs generating "cleaner" distributions than the real one, namely, smaller modes, resulting from noisy labels, are ignored (see Figure 5.1). We define this behavior as *"conditional" mode collapse*, to differentiate it from standard mode collapse, and we analyze it for different GAN objectives (cross-entropy [45], least-squares [99] and Hinge [103]).

A natural application of these findings is the generation of less polluted datasets, *i.e.*, datasets with a lower amount of label noise. We propose to exploit this idea in unsupervised domain adaptation [67, 14, 8, 128, 152], facing the adaptation problem as a *noisy label* problem. More in detail, if we have a model $M_{\theta_s}(x;\theta_s)$, where x is an image and the weights θ_s are trained on labeled samples from the source domain (*e.g.*, via Empirical Risk

Minimization), we can infer a pseudo-label $\tilde{y}$ for each target sample. Typically, due to domain shift [152], a consistent number of labels are wrongly inferred, resulting in a noisy-labelled target set $\{x^{(i)}, \tilde{y}^{(i)}\}_{i=1...m}$.

We propose a training procedure where, at the same time, a classifier can benefit from more and more reliable, conditionally-generated data $\tilde{x}$, while a cGAN can exploit more and more reliable pseudo-labels $\tilde{y}$ inferred by the classifier. In this framework, the source samples are thus exploited only to train an initial classifier M_{θ_s}. After this step, the problem is faced in a fully unsupervised fashion, reducing the noise on the labels of the empirical target distribution over iterations during training. Results on standard unsupervised domain adaptation benchmarks show the effectiveness of our approach.

In summary, the main contributions brought by this Chapter are the following:

1. A thorough study of the conditions in which mode collapse carries benefits, resulting in cGAN models which are more robust to noisy labels, *i.e.*, *"conditional" mode collapse*.

2. The design of a novel training procedure that leverages these findings to generate cleaner data distributions, thus allowing to learn more powerful classifiers.

3. The validation of the proposed algorithm in the unsupervised domain adaptation scenario, which is faced under a completely new perspective with respect to the state of the art. This novel approach shows competitive performance on public benchmarks.

The remaining of this Chapter is organized as follows. In Section 5.2, we detail background and related work. In Section 5.3, we show the robustness of

cGANs to label pollution, through two experiments. Section 5.4 describes how to exploit these findings to tackle unsupervised domain adaptation problems, and Section 5.5 reports the related experimental results. Finally, we draw the conclusions in Section 5.6.

5.2 Connections with related work

In the following, we draw connections between this work and both research related to GANs and to unsupervised domain adaptation.

Generative Adversarial Networks. We remark here the formulation we had introduced in Section 2.4. We had described the original formulation of GANs [45] as a minimax game between a network D (discriminator) and a network G (generator)

$$\min_{\theta_D} \max_{\theta_G} \mathcal{L}_{GAN} = \mathbb{E}_{x \sim p_x}[-\log D(x; \theta_D)] + \mathbb{E}_{z \sim p_z}[-\log(1 - D(G(z; \theta_G); \theta_D))] \tag{5.1}$$

that can be also approached as the alternation between two minimization problems, defined as

$$\min_{\theta_D} \mathcal{L}_D = \mathbb{E}_{x \sim p_x}[-\log D(x; \theta_D)] + \mathbb{E}_{z \sim p_z}[-\log(1 - D(G(z; \theta_G); \theta_D))] \tag{5.2}$$

$$\min_{\theta_G} \mathcal{L}_G = \mathbb{E}_{z \sim p_z}[-\log D(G(z; \theta_G); \theta_D))] \tag{5.3}$$

Solving the optimization problem in Eq. 5.2 makes D classify samples from the data distribution as *real* and samples generated by G as *fake*. Conversely,

solving the optimization problem in Eq. 5.3 makes G generate samples that D would classify as *real*. The generation of new samples happens by feeding G with noise samples $z \sim p_z$.

A straightforward extension is the concatenation of label codes to the input before it is fed to G and D, to condition on the class from which data are generated. This extension is termed *conditional* GAN (cGAN, [102]), and represents the class of models this work focuses on, being our method based on class-conditioned image generation.

Several alternatives to the original GAN formulation [45] have been proposed. Two examples are substituting the cross-entropy loss with the least-squares loss [99] or with the Hinge loss [103]. Also more elaborated alternatives have been introduced [3, 77, 10]. To date, the superiority of one objective function over the others is not fully clear [95], and the main advancements on GAN research have been related to architectural choices [115] and different training procedures [170, 103, 73, 1].

Learning with pseudo-labels. Our training procedure for unsupervised domain adaptation is related to the approach by Lee et al. [65]. In this work, a method for semi-supervised learning is proposed, where, as training proceeds, inference is performed on unlabeled samples, and the "pseudo-labels" obtained are interpreted as correct and used for training a classifier. Part of our method has similarities to this idea since, during our training procedure, we compute pseudo-labels for the target samples. However, we are different in that we use them to train a generative model.

Unsupervised Domain Adaptation. We had introduces the main strategies to face the unsupervised adaptation problem in Section 2.2 and Chapter 4.

Our approach is mostly related to image-to-image translation methods [90,

148, 127, 16, 89, 132, 61], since we exploit generated samples to train a classifier for the target domain. However, we generate these samples through a simple cGAN, *e.g.*, mapping noise vectors and label codes into images. Our architecture is thus much simpler, as we do not need the complex machinery and losses commonly required for translating images.

Our method is substantially different from most unsupervised domain adaptation solutions also because we do not need source samples throughout the adaptation procedure, but only to pre-train the model M_{θ_s}, used to assign pseudo-labels to target samples. Indeed, solutions that align source/target feature statistics [147, 104], map samples from both distributions in a common feature space via adversarial training [39, 40], or translate images between domains [90, 148, 127, 16, 89, 132, 61], are typically based on objectives that depend on both source and target samples. In our case, the independence from source samples during the adaptation procedure brings a number of advantages. The main one is that the training procedure designed for a certain target can be used *as is*, regardless of the source domain, the only difference being the model M_{θ_s} used for the first, initial inference. Moreover, many adaptation methods require additional hyperparameters to balance different loss terms [39, 40, 89, 148, 127, 132, 16] that depend on both source and target samples. The latter is a huge drawback because in unsupervised domain adaptation we do not have target labels for hyperparameter cross-validation.

5.3 Conditional mode collapse

In this section, we study the robustness of cGANs to noisy labels through two experiments, and define the concept of conditional mode collapse. First, we use a cGAN to model a mixture of Gaussians, each associated with a different

class. Second, we train a cGAN on MNIST dataset [82]. In both cases, we vary the amount of noise in the training labels.

Several objectives have been proposed for the GAN formulation, which theoretically minimize different divergences between the data distribution $P_d(x)$ and the generated one $P_g(x)$. For instance, (i) the original GAN, that uses the cross-entropy loss, is proven to minimize the Jensen-Shannon divergence $D_{JS}(p_d||p_g)$ [45]; (ii) the least-squares GAN [99] is proven to minimize the Pearson [112] divergence $D_P(p_d + p_g||2p_g)$; (iii) a GAN with a Hinge loss is proved to minimize the reverse KL divergence $D_{KL}(p_g||p_d)$ [103].

In principle, being the KL divergence not symmetric, minimizing $D_{KL}(p_d||p_g)$ would place high probability everywhere the data occurs, while $D_{KL}(p_g||p_d)$ should enforce low probability wherever the data does *not* occur [44], thus yielding models more prone to mode collapse [47]. Furthermore, it has been suggested that the Pearson divergence is more resistant to outliers than the KL divergence [144, 145], and we can interpret samples with noisy label as outliers in the conditional distributions. We are thus interested in understanding how the different objectives, theoretically associated with different divergences, behave in presence of noisy labels. We aim at answering the following question: *is there a relationship between objectives theoretically prone to mode collapse and GAN models robust to noisy labels?*

5.3.1 Mixture of Gaussians

Consider a dataset composed by a mixture of two one-dimensional Gaussian distributions, representing two classes

$$p_d(x|y=0) = \mathcal{N}(x; \mu_0, \sigma_0) \tag{5.4}$$

$$p_d(x|y=1) = \mathcal{N}(x; \mu_1, \sigma_1) \tag{5.5}$$

Now consider the case where a certain fraction $\alpha < 0.5$ of labels are switched (*e.g.*, Figure 5.1 *(top)*, for $\alpha = 0.1$). The conditional data distribution becomes then

$$p_d(x|y=0) = (1-\alpha)\mathcal{N}(x;\mu_0,\sigma_0) + \alpha\mathcal{N}(x;\mu_1,\sigma_1) \tag{5.6}$$

$$p_d(x|y=1) = (1-\alpha)\mathcal{N}(x;\mu_1,\sigma_1) + \alpha\mathcal{N}(x;\mu_0,\sigma_0) \tag{5.7}$$

We observed that, when modelling such data through a cGAN, the generator fails to recover the secondary, less represented mode (*e.g.*, see Figure 5.1 *(bottom)*).

To provide numerical evidence of what we observed, we present the following experiment. First, we model the distribution defined by Eqs. 5.6 and 5.7 with a cGAN, learning a distribution P_g. Next, we draw samples from each of the two conditional distributions learned by the generator, namely $P_g(x|y=0)$ and $P_g(x|y=1)$, and model them with a Gaussian Mixture model (GM) with two Gaussians:

$$p_{GM}(x) \coloneqq w\,\mathcal{N}(x;\hat{\mu_0},\hat{\sigma_0}) + (1-w)\,\mathcal{N}(x;\hat{\mu_1},\hat{\sigma_1}), \tag{5.8}$$

estimating the mixture parameters via Expectation-Maximization (EM) algorithm. The rationale behind this experiment is that, if a conditional distribution has two modes (one associated with the correct labels, and one associated with the noisy labels), we obtain a two-component GM. If the "noisy" mode is not modeled by $P_g(x|y=i)$, the GM will be defined by a single Gaussian, thus w in Eq. 5.8 is estimated as either 0 or 1 [1].

[1] In practice, we use the function `GaussianMixture` from the Python library `sklearn`, that relies on EM. We initialize it with two Gaussian located in μ_0 and μ_1. If the model only needs 1 (in case the data to model is a single Gaussian), one Gaussian is weighted 0 and the other 1. We observed by modeling the real, noisy conditional distribution that

We propose this analysis for the different objectives described and different levels of noise α, averaging results over 50 runs. The parameters of the original distribution are $\mu_0 = -0.4$, $\mu_1 = 0.4$ and $\sigma_0 = \sigma_1 = 0.05$. The architectures of both generator and discriminator are defined by two 128-dimensional hidden layers. The large number of parameters is to avoid any doubt that the architecture might not have enough capacity to fit the secondary modes.

We report the results in Table 5.1, where we include the estimated parameters for P_{GM} averaged over the different runs. The last column reports the number of times the training of the cGAN resulted in pathological behaviors, that can be summarized in generating data from a single class disregarding the label code provided. This behavior is known as *standard* mode collapse ($s.m.c$), while we are interested in *conditional* mode collapse, defined as mode collapse within a certain class. Actually, we observed that this commonly results in cGANs disregarding, to some extent, samples with noisy labels. The results reported in the other entries of the table are related to runs where the GAN did not evolve in standard mode collapse. We can observe that the cGAN is never able to model the secondary, noise-related mode. Indeed, we observe that both $P_g(x|y = 0)$ and $P_g(x|y = 1)$ are always defined by a single Gaussian instead of a mixture of two (w is always either 0 or 1).

These results suggest thus that cGANs are able to ignore noisy labels to some extent, so as to generate cleaner data. However, as noise α increases, more and more runs result in pathological behaviors. With an amount of noise $\alpha = 0.175$, almost every single training attempt results into a severe standard mode collapsing behavior, where the cGAN ignores the label provided, generating points from a single class. While it is encouraging that

this function can efficiently model two Gaussians if they emerge from the data. We also visually inspected the results associated with every run.

		$p_g(x\|y=0)$		$p_g(x\|y=1)$		
	α	w	$\hat{\mu_0}$	w	$\hat{\mu_1}$	*s.m.c.*
Cross-	.0	1.	-.407 ± .041	0.	.405 ± .041	0/50
Entropy	.05	1.	-.391 ± .029	0.	.399 ± .028	8/50
	.1	1.	-.395 ± .037	0.	.405 ± .039	15/50
	.125	1.	-.399 ± .027	0.	.404 ± .025	6/50
	.15	1.	-.397 ± .014	0.	.398 ± .018	38/50
	.175	1.	-.401 ± .005	0.	.402 ± .002	47/50
Least-	.0	1.	-.414 ± .038	0.	.405 ± .040	0/50
Squares	.05	1.	-.391 ± .034	0.	.408 ± .027	7/50
	.1	1.	-.411 ± .047	0.	.400 ± .035	15/50
	.125	1.	-.387 ± .051	0.	.398 ± .040	17/50
	.15	1.	-.401 ± .030	0.	.392 ± .027	23/50
	.175	1.	-.381 ± .013	0.	.431 ± .041	48/50
Hinge	.0	1.	-.403 ± .039	0.	.395 ± .043	0/50
	.05	1.	-.397 ± .034	0.	.393 ± .030	8/50
	.1	1.	-.408 ± .043	0.	.388 ± .057	9/50
	.125	1.	-.389 ± .022	0.	.404 ± .024	6/50
	.15	1.	-.382 ± .055	0.	.402 ± .022	33/50
	.175	1.	-.439 ± .000	0.	.399 ± .000	49/50

Table 5.1. Results associated with the GMM experiment, averaged over 50 runs. Each row is associated with models trained with different objectives and levels of noise (α). Columns 2 and 3 report the average value of the weight w of equation 5.8 and the average mean of the associated Gaussian, respectively, for the GMM associated with $p_g(x|y=0)$. Columns 4 and 5 report the same analysis for $p_g(x|y=1)$. The last column reports the number of times we observed standard mode collapse (*s.m.c.*).

cGANs can be resistant to label pollution, the last point raised is a concrete issue, being $\alpha = 0.175$ a relatively small amount of noise. Albeit, we would like to stress that standard mode collapse in experiments with Gaussian

mixtures is a well known issue in the GAN literature [101]. As we will detail in following sections, we did not encounter this problem when modeling images, while we did experience the benefits of conditional mode collapse.

Another interesting point of our analysis is the fact that there is not any substantial difference between the different GAN objectives evaluated (cross-entropy [45], least-squares [99], Hinge [103]). The original formulation [45] allows to model the data slightly better (compare how close $\hat{\mu_0}$ and $\hat{\mu_1}$ are to $\mu_0 = -0.4$ and $\mu_1 = 0.4$, respectively), but there are not substantial differences in the way the noise affects the training procedure. This can be related to the fact that GANs are commonly trained by simultaneous gradient descent, which may disregard the original min max game in favour of the (non equivalent) max min problem [47]. Under this light, the divergence that the different objectives aim to minimize is less relevant, and these results are consistent with previous work [95], suggesting that there is no significant difference in the choice of the GAN objective.

5.3.2 Generating cleaner MNIST samples

In this section, we investigate the capability of cGANs to model images under the presence of a fraction of noisy labels α. We consider the MNIST dataset [82] and assume to have an oracle to classify which class a sample belongs to. In practice, this oracle is a CNN [80] trained on MNIST, that achieves $> 99\%$ accuracy on both the training set and the test set. One might genuinely expect that training a cGAN with, *e.g.*, a fraction of noisy labels $\alpha = 0.1$ will result in 10% of mis-generated samples. We show in the following that, in practice, this does not happen. We train the cGAN with different levels of noise α and evaluate the output of the generator through the oracle, by comparing the label code given in input to the cGAN and the output of the

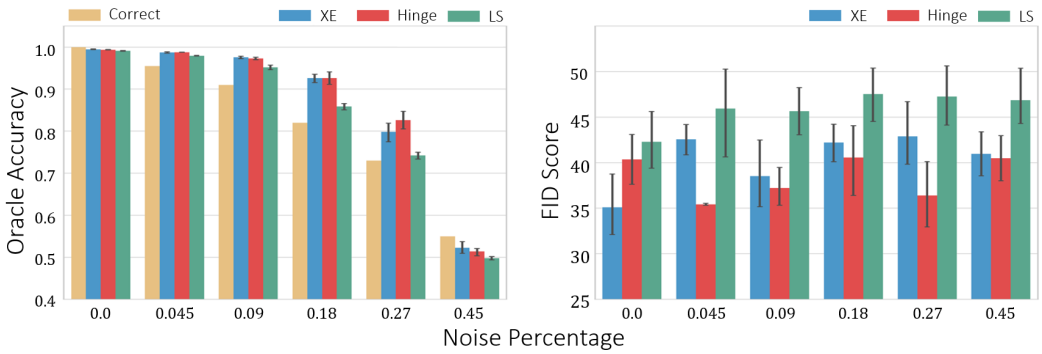


Figure 5.2. *Left* panel shows percentage of images correctly generated by cGANs with different levels α of noise and different objectives (*blue*: cross-entropy [45], *red*: Hinge [103], *green*: least-squares [99]), evaluated through the oracle. *Yellow* bars indicate the percentage $1 - \alpha$ of images with the correct label in the training set. *Right* panel shows the FID scores achieved with different levels of noise and different GAN objectives (same as *left*).

oracle fed with the generated image. Also in this case, we consider different objectives.

Figure 5.2 *(left)* reports our findings. Yellow bars indicate the percentage of correct labels in the dataset $(1 - \alpha)$. Blue, red and green bars indicate the percentage of samples correctly generated by cGANs trained with cross-entropy [45], Hinge [103] and least-squares [99] losses, respectively. As it can be observed, when the level of noise α is reasonably below some threshold, the amount of images correctly generated (*i.e.*, correctly classified by the oracle) is always consistently higher than the amount of clean training samples.

In this experiment, there seem to be some differences between the different objectives, as the least-squares GAN [99] appears to be less resistant to noisy samples. Finally, Figure 5.2 *(right)* reports the FID scores (Fréchet Inception Distance [58]) for the same models. The FID is an indirect measure of image quality, accounting for the distance between the training and the generated distributions (the lower, the better). Interestingly, there seem to be no correlation between the amount of noisy labels and the overall quality

of the images.

5.4 Facing unsupervised domain adaptation

In this section, we detail the method designed to face unsupervised domain adaptation, based on the insights and the findings reported so far.

5.4.1 Method

The pre-train step of our method consists in training a model M_{θ_s} on labeled data from the source distribution

$$\min_{\theta_C} \mathcal{L}_{class} \coloneqq \mathbb{E}_{x,y\sim p_{source}} H(x, y; \theta_s), \tag{5.9}$$

where H is the cross-entropy loss. Equipped with this classifier, we can straightforwardly infer pseudo-labels for the each target sample as $\tilde{y}_t = C(x_t)$. Typically, an unknown percentage of these labels will be wrong, due to the domain shift between P_{source} and P_{target}. We obviously do not know which labels are correct and which are not, but this is irrelevant for the devised strategy.

Before starting the joint training procedure, we need to train a cGAN on the noisy target distribution. This is necessary because we will train C on generated data, and thus starting with randomly-initialized G and D would result in a random classifier C, and consequently in non-informative pseudo-labels.

We train the cGAN in a standard fashion, alternating between the following minimax game. Note that we report here the objective as defined in Goodfellow et al. [45], but in our experiments we also test least-squares GANs [99] and Hinge-GANs [103].

Algorithm 2

Input: target data distribution p_{target}, noise distribution p_z, pre-trained $\theta_D^0, \theta_G^0, \theta_C^0$, step sizes η, δ

Output: learned weights $\theta_D, \theta_G, \theta_C$

1: **Initialize:** $\theta_D \leftarrow \theta_D^0$, $\theta_G \leftarrow \theta_G^0$, $\theta_C \leftarrow \theta_C^0$

2: **while** not done **do**

3: Sample $z \sim p_z$ and $y \sim \Pi_{[0,N]}$

4: Generate $\tilde{x} = G(z|y)$

5: $\theta_C \leftarrow \theta_C - \eta \nabla_{\theta_C} \mathcal{L}_{class}(\theta_C; \tilde{x}, y)$

6: Sample $x \sim p_{target}$ and $z \sim p_z$

7: Infer $\tilde{y} = C(x)$

8: $\theta_D \leftarrow \theta_D - \delta \nabla_{\theta_D} \mathcal{L}_{GAN}(\theta_D; z, x, \tilde{y})$

9: $\theta_G \leftarrow \theta_G - \delta \nabla_{\theta_G} \mathcal{L}_{GAN}(\theta_G; z, \tilde{y})$

$$\min_{\theta_D} \max_{\theta_G} \mathcal{L}_{GAN} \coloneqq \mathbb{E}_{x,\tilde{y}\sim p_{target}}[-\log D(x|\tilde{y})] + \mathbb{E}_{z\sim p_z}[-\log(1 - D(G(z|\tilde{y})|\tilde{y}))] \tag{5.10}$$

In practice, we solve the minimax game via label flipping, dividing it into two minimization problems (as in Eqs 5.2 and 5.3).

Armed with the pre-trained modules C, G and D, we can start the training procedure, which is defined by Algorithm 2. In short, we alternate until convergence between (i) updating the weights of the classifier θ_c via stochastic gradient descent on labeled target samples uniformly generated via G (lines $3-5$), and (ii) training the weights of the discriminator θ_D and of the generator θ_G via stochastic gradient descent on target samples from P_{target}, with pseudo-labels inferred via the classifier C (lines $6-9$). In Figure 5.3,

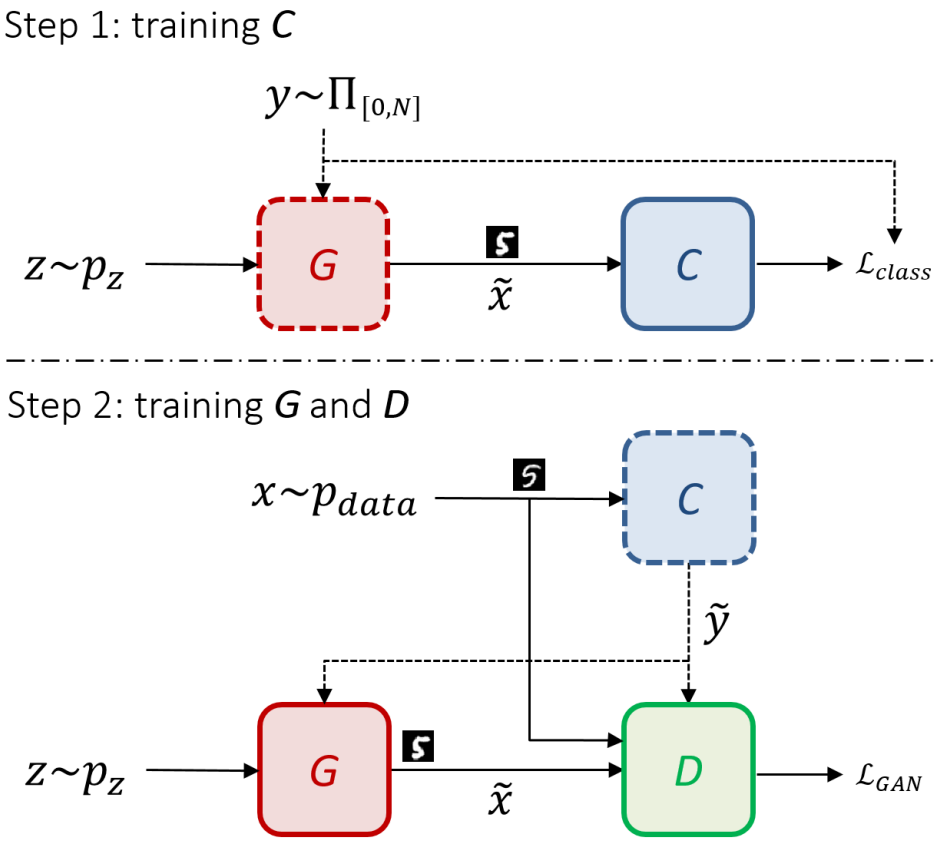


Figure 5.3. Graphical view of Algorithm 2. Step 1 *(top)* and step 2 *(bottom)* refer to lines $3-5$ and $6-9$, respectively. The module G and the module D are the generator and the discriminator of the cGAN. The module C is the classifier. Dashed boxes indicate frozen modules (not trained). Solid and dashed wires indicate image and label flows, respectively.

we report a graphical view of the information flow of the proposed system: *top* (step 1) and *bottom* (step 2) panels represent modules that concern lines $3-5$ and $6-9$ of Algorithm 2, respectively.

The output of Algorithm 2 is twofold: (i) the trained modules of the cGAN (G and D), and (ii) the trained classifier C, which is the module finally used to classify target samples. In the next section, we report performances obtained using C in unsupervised domain adaptation benchmarks.

5.5 Experiments

We test Algorithm 2 on a variety of unsupervised domain adaptation benchmarks. In all experiments, we run Algorithm 2 until convergence, intended as convergence of the cGAN minimax game, and use accuracy on target dataset test sets (fed to the classifier C) as a metric to evaluate our models and compare them with other adaptation approaches.

Benchmarks. We test our method on the following cross-dataset digit classification problems: SVHN $\leftrightarrow$ MNIST, MNIST $\rightarrow$ MNIST-M and USPS $\leftrightarrow$ MNIST, following protocols on which unsupervised domain adaptation algorithms based on GANs are tested [90, 148, 16, 127, 132]. In order to work with comparable sizes, we resized all images to 32×32. For each experiment, we use a CNN [80] with architecture *conv-pool-conv-pool-fc-fc-softmax*. For the GAN architectures, we draw inspiration from DCGAN [115], though considering different objectives (cross-entropy, least-squares, Hinge). All details regarding architectures and training procedures are reported in C.

5.5.1 Results

We report in Figure 5.4 the plots showing the evolution of test accuracy in different experiments throughout the training procedure defined by Algorithm 2. As can be observed, the classifier trained on target samples generated via the cGAN performs iteratively better on the target distribution of interest. It is worth highlighting the monotonic increase of performance: early stopping is not feasible in unsupervised domain adaptation, thus an unstable algorithm is of scarce utility.

A particularly important result in the one related to the MNIST $\rightarrow$ SVHN split. The large gap between the two domains, and the fact that labels are

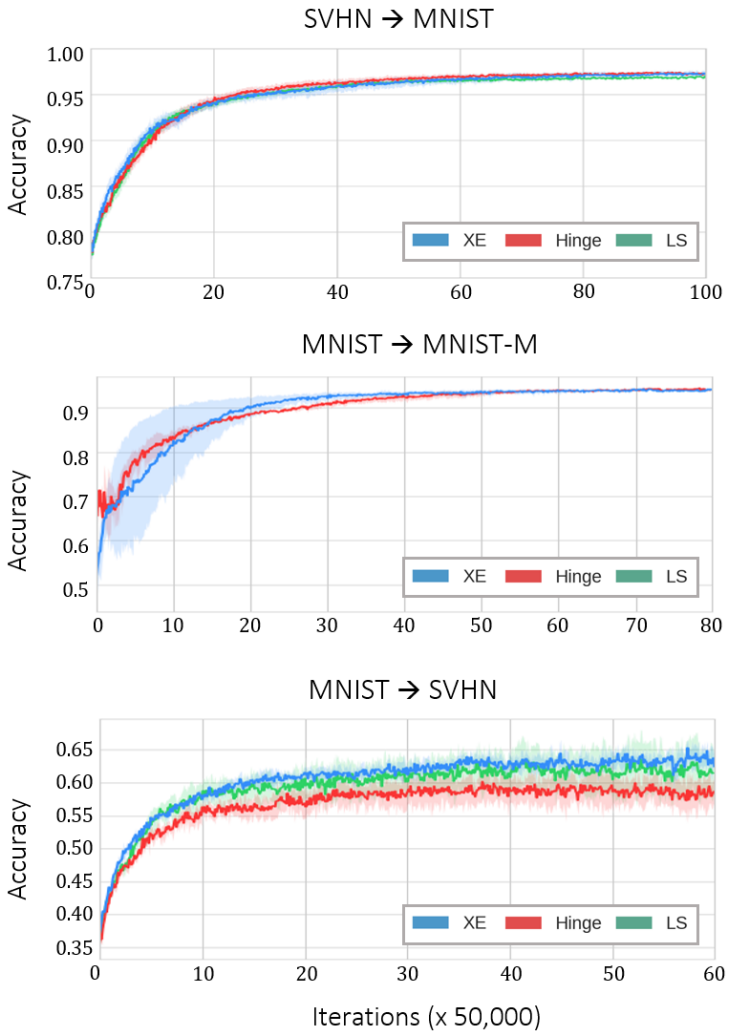


Figure 5.4. Evolution of the accuracies on target test sets for SVHN → MNIST, MNIST → MNIST-M and MNIST → SVHN (from *left* to *right*), computed throughout the training procedure described in Algorithm 2. *Blue*, *red* and *green* are associated with GANs trained with the cross-entropy loss [45], the Hinge loss [103] and the least-squares loss [99], respectively. Results obtained with the least-squares loss are not reported for the MNIST → MNIST-M as they are significantly worse than the ones achieved with the other options. Curves were averaged over three different runs, shades represent the confidence bands.

provided for the easier, more biased dataset makes this split particularly difficult to tackle [39, 40]. Our method allows to generate SVHN samples that

make the classifier C – trained on them – better generalizing to the target distribution, improving performance of $\sim 30\%$ with respect to the baseline. The complete analysis of the obtained results, also in comparison with the state-of-the-art methods, is illustrated in the following. Images generated can be found at the end of the Chapter (Figures 5.6, 5.7, 5.8, 5.9, 5.10).

Comparison with other methods. Table 5.2 and 5.3 compare the proposed method performance with the results obtained by several works in the literature. It is worth to note that, nowadays, research in unsupervised domain adaptation reached a point where it is difficult to state the superiority of a method over the others. Indeed, Tables 5.2 and 5.3 show that there is not a single method that performs better than the others in *every* benchmark. In the following, we discuss in which scenarios our method is a better option with respect to others, and in which it might not.

First, our method shows performance comparable with the state of the art in the SVHN $\rightarrow$ MNIST split benchmark (see Table 5.2), significantly outperforming more complex image-to-image translation methods [148, 89, 127, 132] that not only rely on more complicated architectures, but also present a training procedure where the objective is weighted by different hyperparameters (which, as previously mentioned, is a significant drawback in unsupervised domain adaptation). Next, an important result is the one related to MNIST $\rightarrow$ SVHN (see Table 5.2). As discussed above, this is a rather challenging split, and several methods (*e.g.*, [39, 153, 158, 104, 90, 89, 148]) do not show results on this benchmark. Our algorithm, with the cross-entropy loss as GAN objective, is the best performing method by a statistically significant margin. On MNIST $\rightarrow$ MNIST-M (see Table 5.2), the performance achieved with our method is comparable with Saito et al. [129] and below the one achieved by methods that perform hyperparameter cross-validation

Source	**SVHN**	**MNIST**	**MNIST**
Target	**MNIST**	**SVHN**	**MNIST-M**
Train on source	0.682	0.0 ± 0.0	0.0 ± 0.0
DSN [17]	0.829	-	0.832
DIFA [158]	0.897 ± 0.020	-	-
MECA [104]	0.952	-	-
ATT [129]	0.862	0.528	0.942
AD [130]	0.950 ± 0.187	-	-
MCD [131]	0.962 ± 0.004	-	-
DTN* [148]	0.849	-	-
UNIT* [89]	0.905	-	-
PixelDA** [16]	-	-	0.982
SBADA** [127]	0.761	0.611	0.994
CycADA [61]	0.904 ± 0.004	-	-
Ours			
Cross-entropy	0.973 ± 0.006	0.634 ± 0.026	0.943 ± 0.002
Least-squares	0.969 ± 0.003	0.618 ± 0.060	-
Hinge	0.973 ± 0.003	0.586 ± 0.041	0.938 ± 0.002
Train on target	0.992	0.0 ± 0.0	0.0 ± 0.0

Table 5.2. Comparison of our method with different objectives (cross-entropy, least-squares and Hinge) with competing algorithms. Results averaged over 3 different runs. (*) Uses extra SVHN data ($531,131$ images). (**) Uses $1,000$ target samples for cross-validation.

[16, 127].

The benchmarks USPS $\leftrightarrow$ MNIST (see Table 5.3) represent more problematic setups for our method. Indeed, it could happen that the first model M_{θ_s} completely misclassifies samples from one class (in rare cases, from 2 classes).

Source	**USPS**	**MNIST**
Target	**MNIST**	**USPS**
Train on source	0.627	0.797
DIFA [158]	0.897 ± 0.005	0.962 ± 0.002
AD [130]	0.931 ± 0.127	0.961 ± 0.029
MCD [131]	0.941 ± 0.003	0.965 ± 0.003
CoGAN [90]	0.931	0.957
UNIT* [89]	0.936	0.960
PixelDA** [16]	-	0.959
SBADA** [127]	0.950	0.976
CycADA [61]	0.965 ± 0.001	0.956 ± 0.002
Ours		
Cross-entropy	0.918 ± 0.013	0.893 ± 0.019
Least-squares	0.916 ± 0.019	0.903 ± 0.013
Hinge	0.891 ± 0.010	0.907 ± 0.022
Train on target	0.992	0.999

Table 5.3. Comparison of our method with different objectives (cross-entropy, least-squares and Hinge) with competing algorithms. Results averaged over 3 different runs. We use the whole source training sets in both splits. (*) Uses extra SVHN data ($531,131$ images). (**) Uses $1,000$ target samples for cross-validation.

In these circumstances, the pseudo-labels used are misleading, and the cGAN will confuse some classes. This drawback might make our algorithm a worse option with respect to other algorithms [104, 130, 131]. We argue though that this limitation could likely be present in several other algorithms that perform feature confusion [39, 153, 158]. Indeed, these algorithms force target samples in a feature space that resembles the source one, but we do not have

any information regarding the *correctness* of the regions of the space in which the points are mapped.

5.5.2 Limitations, negative results and future work

As introduced in the previous section, the main limitation of our method occurs when the classifier M_{θ_s} provides initial pseudo-labels for the target samples with a too "structured" noise, *e.g.*, when the classifier always mistakes samples from a certain class. For example, Figure 5.5 *(bottom)* reports a pathological experiment where the original classifier trained on USPS almost never predicts MNIST samples from the class '9' with the correct label (Figure 5.5, *bottom-left*). This results in a cGAN misled by the uncorrectness of the pseudo-labels, and thus in a final classifier that always confuses two classes (Figure 5.5, *bottom-right*). In this particular example, 9 and 7 classes are almost completely interchanged.

A limitation is also represented by the current difficulty of generating high-resolution images with GANs. Indeed, current methods for generating high-resolution, realistic samples require access to a huge amount of resources [1]. This is the reason for which we did not include results associated with the Office-31 dataset [128] and the VisDA 2017 challenge [113]. Limitations of GAN-based methods for unsupervised domain adaptation are also discussed in Russo et al. [127], and we come out to similar conclusions.

Nevertheless, an upside of our method is that it generates target samples by relying on standard cGANs, without the need of image-to-image translation. For this reason, state-of-the-art GAN architectures can be embedded in our pipeline without additional effort. This is not the case for image-to-image translation methods, which rely on generally more complex architectures and losses. We leave the experiments regarding the generation of higher-resolution

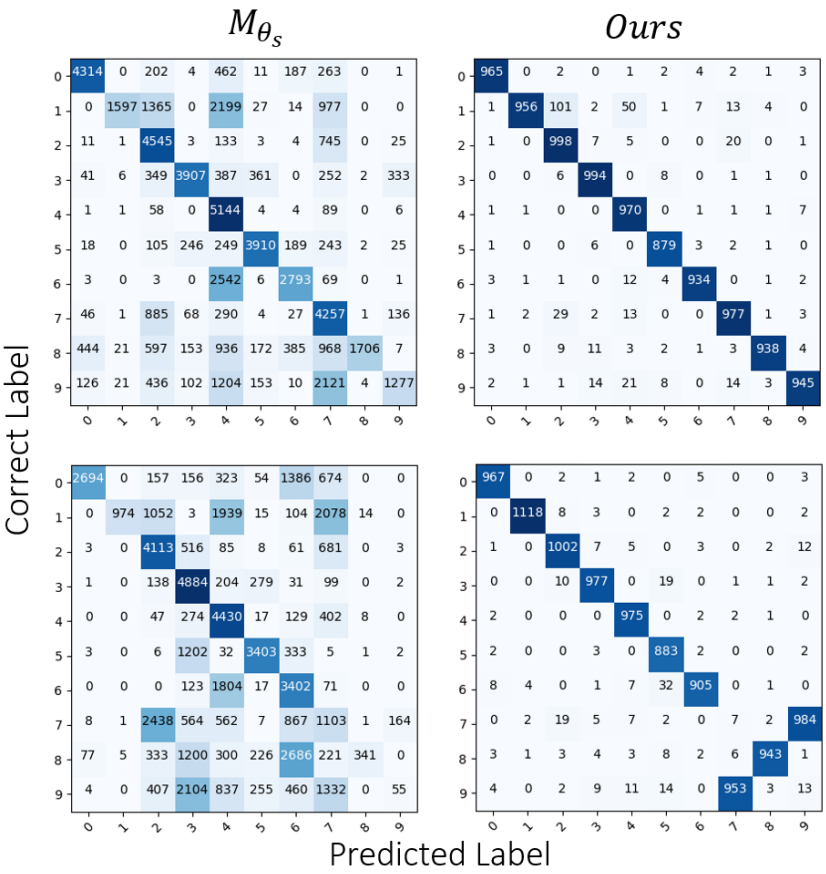


Figure 5.5. Confusion matrices of models trained on USPS and tested on MNIST. *Left* panels report performance related to initial model trained on source samples (M_{θ_s}). *Right* panels report performance related to models trained through Algorithm 2. The *top* panels report a case where the initial performance is good enough to allow good classification with our method (see *top-right* panel). The *bottom* panels report an example where—after our training procedure—samples related to classes 7 and 9 result mis-classified due to the bad initial condition.

images for future work. It is encouraging though that our pipeline works on MNIST $\rightarrow$ SVHN, in that SVHN samples, even though at low-resolution, are fully realistic images.

5.6 Conclusions

We introduce the concept of "conditional" mode collapse, which makes cGANs resistant to label noise, and carry out a thorough analysis on different GAN objectives. We design a training procedure that allows to generate cleaner data distributions, and propose unsupervised domain adaptation as applicative setting. We show that it is possible to generate cleaner samples from the target distribution, and that models trained on them perform comparably or better than the state of the art in different benchmarks.

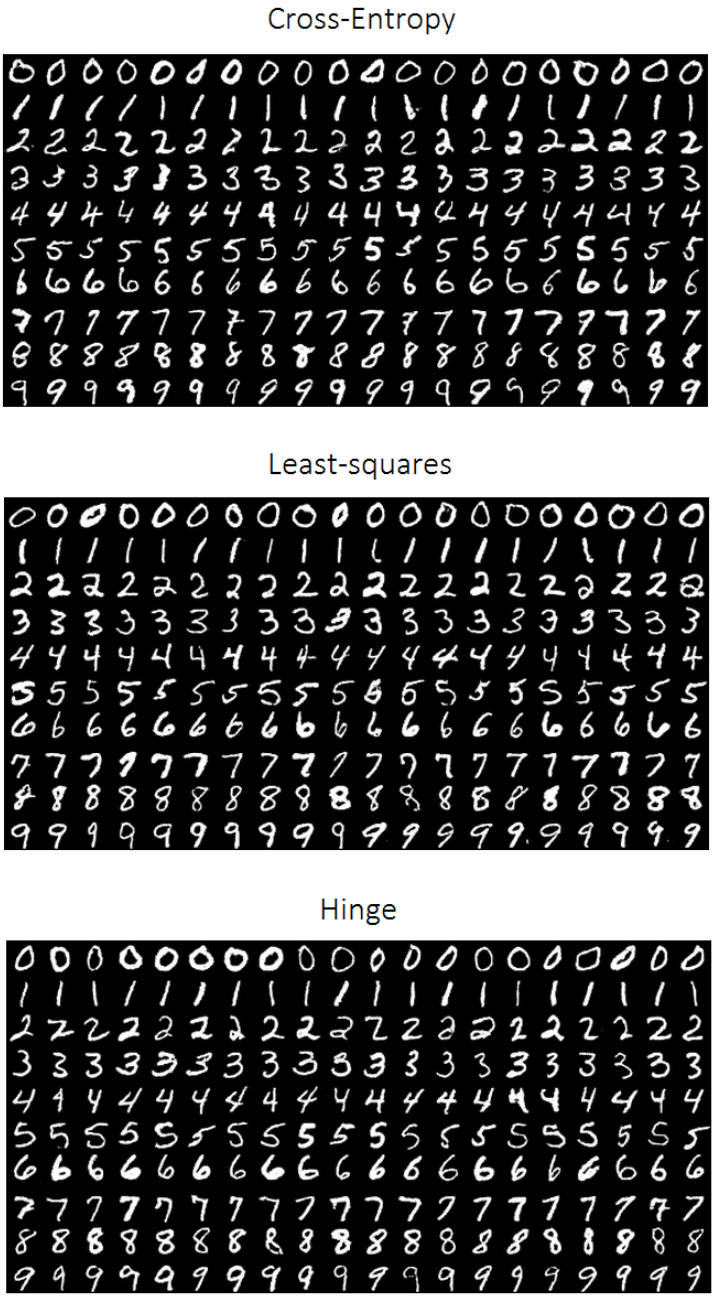


Figure 5.6. MNIST samples generated by G, trained with Algorithm 2 (SVHN $\rightarrow$ MNIST split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).

Figure 5.7. SVHN samples generated by G, trained with Algorithm 2 (MNIST $\rightarrow$ SVHN split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).

Figure 5.8. MNIST-M samples generated by G, trained with Algorithm 2 (MNIST $\rightarrow$ MNIST-M split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).

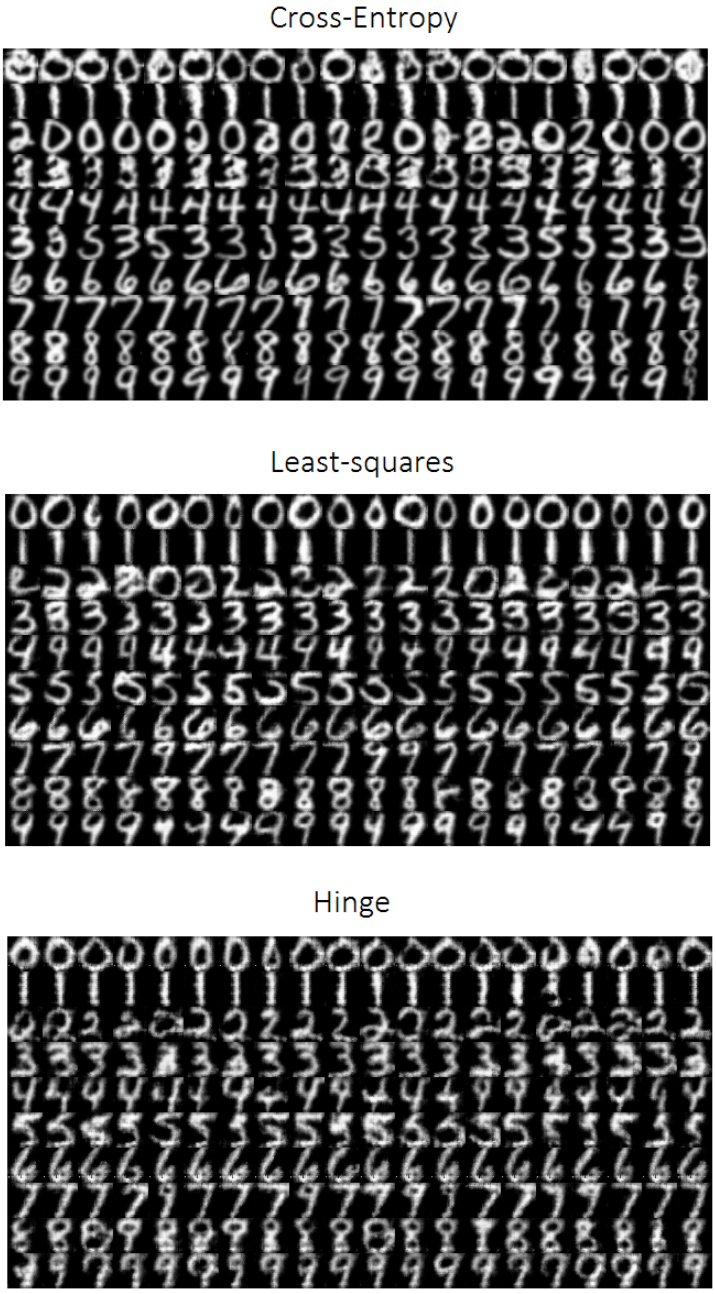


Figure 5.9. USPS samples generated by G, trained with Algorithm 2 (MNIST $\rightarrow$ USPS split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).

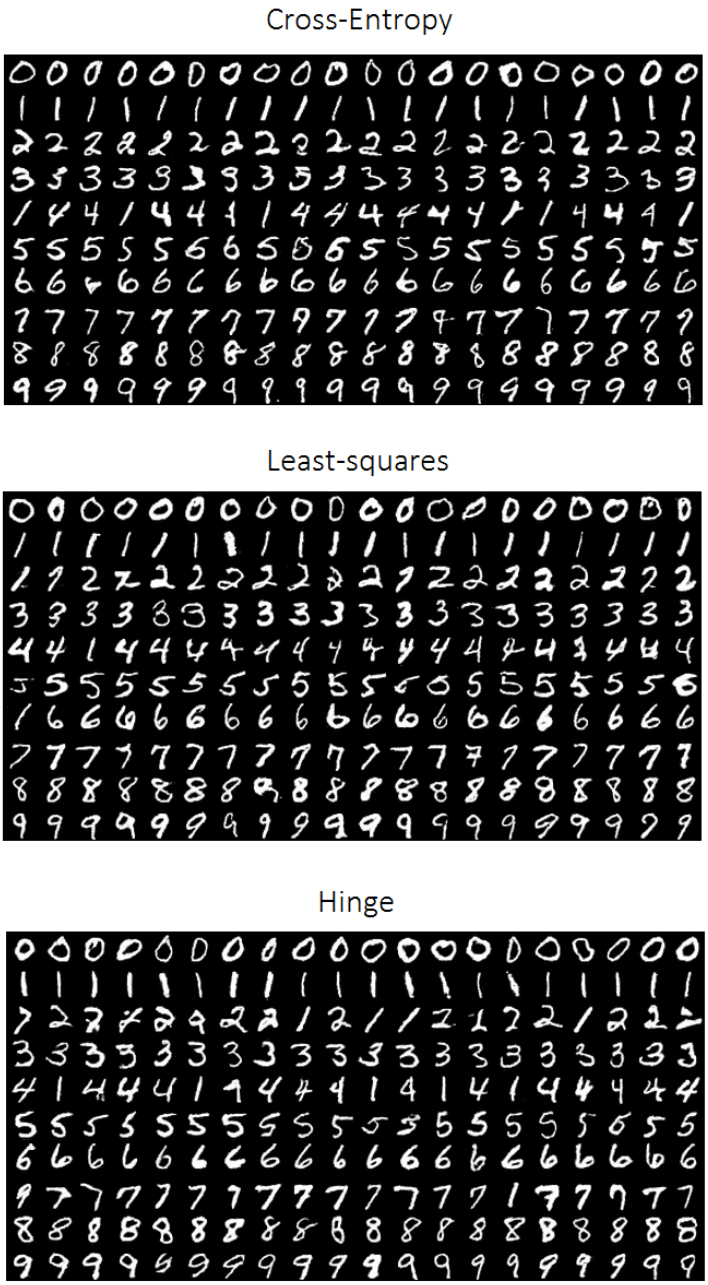


Figure 5.10. MNIST samples generated by G, trained with Algorithm 2 (USPS $\rightarrow$ MNIST split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).

Chapter 6

Generalizing to Unseen Domains

6.1 A new framework to study generalization

In many modern applications of machine learning, we wish to learn a system that can perform uniformly well across multiple populations. Due to high costs of data acquisition, however, it is often the case that datasets consist of a limited number of population sources. While performance evaluated on the validation dataset—usually from the same population as the training dataset—is a standard metric on which many systems are optimized, it has been observed that performance on populations different from that of the training data can be much worse [67, 14, 8, 128, 152]. In this Chapter, we are concerned with generalizing to populations different from the training distribution, in settings where we have no access to any data from the unknown target distributions.

As we had discussed in Chapter 2, a number of authors have proposed *domain adaptation* methods (*e.g.*, [39, 153, 147, 104, 158]) in settings where

a fully labeled source dataset and an unlabeled (or partially labeled) set of examples from fixed target distributions are available, and *domain generalization* methods (*e.g.*, [74, 165, 107, 41, 85, 134, 96, 98, 86]), in settings where the training distribution is defined by different sources.

In this work, we develop methods that can learn to better generalize to new unknown domains, under the restrictive setting where training data only comes from a *single* source domain, and we do not have any prior knowledge on the target distributions. For example, consider a module for self-driving cars that needs to generalize well across different cities unexplored during training, and the source distribution is constituted by a single city.

Inspired by recent developments in distributionally robust optimization and adversarial training [139, 83, 56], we consider the following worst-case problem around the (training) source distribution P_0

$$\underset{\theta \in \Theta}{\text{minimize}} \sup_{P : D(P, P_0) \leq \rho} \mathbb{E}_P[\ell(\theta; (X, Y))]. \tag{6.1}$$

Here, $\theta \in \Theta$ is the model, $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ is a source data point with its labeling, $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is the loss function, and $D(P, Q)$ is a distance metric on the space of probability distributions.

The solution to worst-case problem (6.1) guarantees good performance against data distributions that are distance ρ away from the source domain P_0. To allow data distributions that have different support to that of the source P_0, we use Wasserstein distances as our metric D. Our distance will be defined on the semantic space, so that target populations P satisfying $D(P, P_0) \leq \rho$ will be realistic covariate shifts that preserve the same semantic representation of the source (*e.g.*, adding color to a greyscale image). In this regard, we expect the solution to the worst-case problem (6.1)—the model that we wish to learn—to have favorable performance across covariate shifts

in the semantic space. By *semantic space* we mean learned representations, since recent works [31, 70] suggest that distances in the space of learned representations of high capacity models typically correspond to semantic distances in visual space

We propose an iterative procedure that aims to solve the problem (6.1) for a small value of ρ at a time, and does stochastic gradient updates to the model θ with respect to these fictitious worst-case target distributions (Section 6.2). Each iteration of our method uses small values of ρ, and we provide a number of theoretical interpretations of our method. First, we show that our iterative algorithm is an adaptive data augmentation method where we add adversarially perturbed samples—at the current model—to the dataset (Section 6.3). More precisely, our adversarially generated samples roughly correspond to *Tikhonov regularized Newton-steps* [84, 100] on the loss in the semantic space. Further, we show that for softmax losses, each iteration of our method can be thought of as a data-dependent regularization scheme where we regularize towards the parameter vector corresponding to the true label, instead of regularizing towards zero like classical regularizers such as ridge or lasso.

From a practical viewpoint, a key difficulty in applying the worst-case formulation (6.1) is that the magnitude of the covariate shift ρ is a priori unknown. We propose to learn an ensemble of models that correspond to different distances ρ. In other words, our iterative method generates a collection of datasets, each corresponding to a different inter-dataset distance level ρ, and we learn a model for each of them. At test time, we use a heuristic method to choose an appropriate model from the ensemble.

We test our approaches on a simple digit recognition task, and a more realistic semantic segmentation task across different seasons and weather

conditions. In both settings, we observe that our method allows to learn models that improve performance across a priori unknown target distributions that have varying distance from the original source domain.

Related work

The literature on adversarial training [46, 139, 83, 56] is closely related to our work, since the main goal is to devise training procedures that learn models robust to fluctuations in the input. Departing from imperceptible attacks considered in adversarial training, we aim to learn models that are resistant to larger perturbations, namely out-of-distribution samples. Sinha et al. [139] propose a principled adversarial training procedure, where new images that maximize some risk are generated and the model parameters are optimized with respect to those adversarial images. Being devised for defense against *imperceptible* adversarial attacks, the new images are learned with a loss that penalizes differences between the original and the new ones. In this work, we rely on a minimax game similar to the one proposed by Sinha et al. [139], but we impose the constraint in the semantic space, in order to allow our adversarial samples from a fictitious distribution to be different at the pixel level, while sharing the same semantics.

The body of work on domain adaptation and domain generalization are related to our work, but, as discussed, we consider the more general setting where training data comes form a single source distribution and we do not have information regarding the target distributions on which the system will be deployed. One could term our approach as "single-source" or "unsupervised" domain generalization.

Tobin et al. [151] propose *domain randomization*, which applies to simulated data and creates a variety of random renderings with the simulator,

hoping that the real world will be interpreted as one of them. Our goal is the same, since we aim at obtaining data distributions more similar to the real world ones, but we accomplish it by actually *learning* new data points, and thus making our approach applicable to any data source and without the need of a simulator.

Hendrycks and Gimpel [57] suggest that a good empirical way to detect whether a test sample is out-of-distribution for a given model is to evaluate the statistics of the softmax outputs. We adapt this idea in our setting, learning ensemble of models trained with our method and choosing at test time the model with the greatest maximum softmax value.

6.2 Adversarial data augmentation

The worst-case formulation (6.1) over domains around the source P_0 hinges on the notion of distance $D(Q, P_0)$, that characterizes the set of unknown populations we wish to generalize to. Conventional notions of Wasserstein distance used for adversarial training [139] are defined with respect to the original input space $\mathcal{X}$, which for images corresponds to raw pixels. Since our goal is to consider fictitious target distributions corresponding to realistic covariate shifts, we define our distance on the semantic space. Before properly defining our setup, we first give a few notations. Letting p be the dimension of output of the last hidden layer, we denote $\theta = (\theta_c, \theta_f)$ where $\theta_c \in \mathbb{R}^{p \times m}$ is the set of weights of the final layer, and θ_f is the rest of the weights of the network. We denote by $g(\theta_f; x)$ the output of the embedding layer of our neural network. For example, in the classification setting, m is the number of classes and we consider the softmax loss

$$\ell(\theta; (x, y)) := -\log \frac{\exp\left(\theta_{c,y}^\top g(\theta_f; x)\right)}{\sum_{j=1}^m \exp\left(\theta_{c,j}^\top g(\theta_f; x)\right)} \tag{6.2}$$

where $\theta_{c,j}$ is the j-th column of the classification layer weights $\theta_c \in \mathbb{R}^{p\times m}$.

Wasserstein distance on the semantic space. On the space $\mathbb{R}^p \times \mathcal{Y}$, consider the following transportation cost c—cost of moving mass from (z, y) to (z', y')

$$c((z, y), (z', y')) := \frac{1}{2} \|z - z'\|_2^2 + \infty \cdot \mathbf{1}\{y \neq y'\}.$$

The transportation cost takes value ∞ for data points with different labels, since we are only interested in perturbation to the marginal distribution of X. We now define our notion of distance on the semantic space. For inputs coming from the original space $\mathcal{X} \times \mathcal{Y}$, we consider the transportation cost c_θ defined with respect to the output of the last hidden layer

$$c_\theta((x, y), (x', y')) := c((g(\theta_f; x), y), (g(\theta_f; x'), y'))$$

so that c_θ measures distance with respect to the feature mapping $g(\theta_f; x)$. For probability measures P and Q both supported on $\mathcal{X} \times \mathcal{Y}$, let $\Pi(P, Q)$ denote their couplings, meaning measures M with $M(A, \mathcal{X} \times \mathcal{Y}) = P(A)$ and $M(\mathcal{X} \times \mathcal{Y}, A) = Q(A)$. Then, we define our notion of distance by

$$D_\theta(P, Q) := \inf_{M\in\Pi(P,Q)} \mathbb{E}_M[c_\theta((X, Y), (X', Y'))]. \tag{6.3}$$

Armed with this notion of distance on the semantic space, we now consider a variant of the worst-case problem (6.1) where we replace the distance with D_θ (6.3), our adaptive notion of distance defined on the semantic space

$$\underset{\theta\in\Theta}{\text{minimize}} \ \sup_P \{\mathbb{E}_P[\ell(\theta; (X, Y))] : D_\theta(P, P_0) \leq \rho\}.$$

Computationally, the above supremum over probability distributions is intractable. Hence, we consider the following Lagrangian relaxation with penalty parameter γ

$$\underset{\theta\in\Theta}{\text{minimize}} \ \sup_P \{\mathbb{E}_P[\ell(\theta; (X, Y))] - \gamma D_\theta(P, P_0)\}. \tag{6.4}$$

Taking the dual reformulation of the penalty relaxation (6.4), we can obtain an efficient solution procedure. The following result is a minor adaptation of [13, Theorem 1]; to ease notation, let us define the robust surrogate loss

$$\phi_\gamma(\theta;(x_0,y_0)) := \sup_{x\in\mathcal{X}} \left\{\ell(\theta;(x,y_0)) - \gamma c_\theta((x,y_0),(x_0,y_0))\right\}. \tag{6.5}$$

Lemma 1. *Let $\ell : \Theta \times (\mathcal{X}\times\mathcal{Y}) \to \mathbb{R}$ be continuous. For any distribution Q and any $\gamma \geq 0$, we have*

$$\sup_P \left\{\mathbb{E}_P[\ell(\theta;(X,Y))] - \gamma D_\theta(P,Q)\right\} = \mathbb{E}_Q[\phi_\gamma(\theta;(X,Y))]. \tag{6.6}$$

In order to solve the penalty problem (6.4), we can now perform stochastic gradient descent procedures on the robust surrogate loss ϕ_γ. Under suitable conditions [19], we have

$$\nabla_\theta \phi_\gamma(\theta;(x_0,y_0)) = \nabla_\theta \ell(\theta;(x_\gamma^\star,y_0)), \tag{6.7}$$

where $x_\gamma^\star = \arg\max_{x\in\mathcal{X}} \left\{\ell(\theta;(x,y_0)) - \gamma c_\theta((x,y_0),(x_0,y_0))\right\}$ is an adversarial perturbation of x_0 at the current model θ. Hence, computing gradients of the robust surrogate ϕ_γ requires solving the maximization problem (6.5). Below, we consider an (heuristic) iterative procedure that iteratively performs stochastic gradient steps on the robust surrogate ϕ_γ.

Iterative Procedure. We propose an iterative training procedure where two phases are alternated: a *maximization* phase where new data points are learned by computing the inner maximization problem (6.5) and a *minimization* phase, where the model parameters are updated according to stochastic gradients of the loss evaluated on the adversarial examples generated from the maximization phase. The latter step is equivalent to stochastic gradient steps on the robust surrogate loss ϕ_γ, which motivates its name. The main idea here

is to iteratively learn "hard" data points from fictitious target distributions, while preserving the semantic features of the original data points.

Concretely, in the k-th *maximization* phase, we compute n adversarially perturbed samples at the current model $\theta \in \Theta$

$$X_i^k \in \arg\max_{x \in \mathcal{X}} \left\{ \ell(\theta; (x, Y_i)) - \gamma c_\theta((x, Y_i), (X_i^{k-1}, Y_i)) \right\} \tag{6.8}$$

where X_i^0 are the original samples from the source distribution P_0. The *minimization phase* then performs repeated stochastic gradient steps on the augmented dataset $\{X_i^k, Y_i\}_{0 \le k \le K, 1 \le i \le n\}}$. The maximization phase (6.8) can be efficiently computed for smooth losses if $x \mapsto c_{\theta^{k-1}}((x, Y_i), (X_i^{k-1}, Y_i))$ is strongly convex [139, Theorem 2]; for example, this is provably true for any linear network. In practice, we use gradient ascent steps to solve for worst-case examples (6.8); see Algorithm 3 for the full description of our algorithm.

Ensembles for classification. The hyperparameter γ—which is inversely proportional to ρ, the distance between the fictitious target distribution and the source—controls the ability to generalize outside the source domain. Since target domains are unknown, it is difficult to choose an appropriate level of γ a priori. We propose a heuristic ensemble approach where we train s models $\{\theta^0, ..., \theta^s\}$. Each model is associated with a different value of γ, and thus to fictitious target distributions with varying distances from the source P_0. To select the best model at test time—inspired by Hendrycks and Gimpel [57]—given a sample x, we select the model $\theta^{u^\star(x)}$ with the greatest softmax score

$$u^\star(x) := \arg\max_{1 \le u \le s} \max_{1 \le j \le k} \theta_{c,j}^{u\top} g(\theta_f^u; x). \tag{6.9}$$

Algorithm 3 Adversarial Data Augmentation

Input: original dataset $\{X_i, Y_i\}_{i=1,...,n}$ and initialized weights θ_0

Output: learned weights θ

Initialize: $\theta \leftarrow \theta_0$
for $k = 1, ..., K$ **do** ▷ Run the minimax procedure K times
  for $t = 1, ..., T_{\min}$ **do**
    Sample (X_t, Y_t) uniformly from dataset
    $\theta \leftarrow \theta - \alpha \nabla_\theta \ell(\theta; (X_t, Y_t))$
  Sample $\{X_i, Y_i\}_{i=1,...,n}$ uniformly from the dataset
  for $i = 1, \ldots, n$ **do**
    $X_i^k \leftarrow X_i$
    for $t = 1, \ldots, T_{\max}$ **do**
      $X_i^k \leftarrow X_i^k + \eta \nabla_x \left\{ \ell(\theta; (X_i^k, Y_i)) - \gamma c_\theta((X_i^k, Y_i), (X_i, Y_i)) \right\}$
    Append (X_i^k, Y_i^k) to dataset
for $t = 1, \ldots, T$ **do**
  Sample (X, Y) uniformly from dataset
  $\theta \leftarrow \theta - \alpha \nabla_\theta \ell(\theta; (X, Y))$

6.3 Theoretical motivation

In our iterative algorithm (Algorithm 3), the *maximization* phase (6.8) was a key step that augmented the dataset with adversarially perturbed data points, which was followed by standard stochastic gradient updates to the model parameters. In this section, we provide some theoretical understanding of the augmentation step (6.8). First, we show that the augmented data points (6.8) can be interpreted as *Tikhonov regularized Newton-steps* [84, 100] on the loss in the semantic space (under the current model). Roughly speaking, this

quantifies the sense in which Algorithm 3 is an adaptive data augmentation algorithm that adds data points from fictitious "hard" target distributions. Secondly, recall that the robust surrogate (6.5) is the loss whose stochastic gradients were used to update the model parameters θ in the *minimization* step (Eq (6.7)). In the classification setting, we show that the robust surrogate (6.5) roughly corresponds to a novel data-dependent regularization scheme on the softmax loss ℓ. Instead of penalizing towards zero like classical regularizers (*e.g.*, ridge or lasso), our data-dependent regularization term penalizes deviations from the parameter vector corresponding to that of the true label.

6.3.1 Adaptive data augmentation

We now give an interpretation for the augmented data points in the maximization phase (6.8). Concretely, we fix $\theta \in \Theta$, $x_0 \in \mathcal{X}$, $y_0 \in \mathcal{Y}$, and consider an ϵ-maximizer

$$x_\epsilon^\star \in \epsilon\text{-}\arg\max_{x \in \mathcal{X}} \left\{ \ell(\theta; (x, y_0)) - \gamma c_\theta((x, y_0), (x_0, y_0)) \right\}.$$

We let $z_0 := g(\theta_f; x_0) \in \mathbb{R}^p$, and abuse notation by using $\ell(\theta; (z_0, y_0)) := \ell(\theta; (x_0, y_0))$. In what follows, we show that the feature mapping $g(\theta_f; x_\epsilon^\star)$ satisfies

$$g(\theta_f; x_\epsilon^\star) = \underbrace{g(\theta_f; x_0) + \frac{1}{\gamma}\left(I - \frac{1}{\gamma}\nabla_{zz}\ell(\theta; (z_0, y_0))\right)^{-1} \nabla_z \ell(\theta; (z_0, y_0))}_{=:\ \widehat{g}_{\text{newton}}(\theta_f; x_0)} + O\left(\frac{\epsilon}{\gamma} + \frac{1}{\gamma^4}\right). \tag{6.10}$$

Intuitively, this implies that the adversarially perturbed sample $x_\epsilon^\star$ is drawn from a fictitious target distribution where probability mass on $z_0 = g(\theta_f; x_0)$ was transported to $\widehat{g}_{\text{newton}}(\theta_f; x_0)$. We note that the transported point in the semantic space corresponds to a *Tikhonov regularized Newton-step* [84, 100]

on the loss $z \mapsto \ell(\theta; (z, y_0))$ at the current model θ. Noting that computing $\widehat{g}_{\text{newton}}(\theta_f; x_0)$ involves backsolves on a large dense matrix, we can interpret our gradient ascent updates in the maximization phase (6.8) as an iterative scheme for approximating this quantity.

We assume sufficient smoothness, where we use $\|H\|$ to denote the ℓ_2-operator norm of a matrix H.

Assumption 1. *There exists $L_0, L_1 > 0$ such that, for all $z, z' \in \mathbb{R}^p$, we have $|\ell(\theta; (z, y_0)) - \ell(\theta; (z', y_0))| \leq L_0 \|z - z'\|_2$ and $\|\nabla_z \ell(\theta; (z, y_0)) - \nabla_z \ell(\theta; (z', y_0))\|_2 \leq L_1 \|z - z'\|_2$.*

Assumption 2. *There exists $L_2 > 0$ such that, for all $z, z' \in \mathbb{R}^p$, we have $\|\nabla_{zz} \ell(\theta; (z, y_0)) - \nabla_{zz} \ell(\theta; (z', y_0))\| \leq L_2 \|z - z'\|_2$.*

Then, we have the following bound (6.10) whose proof we defer to Appendix D.

Theorem 1. *Let Assumptions 1, 2 hold. If $Im(g(\theta_f; \cdot)) = \mathbb{R}^p$ and $\gamma > L_1$, then*

$$\|g(\theta_f; x_\epsilon^\star) - \widehat{g}_{\text{newton}}(\theta_f; x_0)\|_2 \leq \frac{2\epsilon}{\gamma - L_1} + \frac{L_2}{3(\gamma - L_1)} \left\{ 4 \left(\frac{5L_0}{\gamma} \right)^3 + \left(\frac{L_0}{\gamma - L_1} \right)^3 + \left(\frac{2\epsilon}{\gamma} \right)^{\frac{3}{2}} \right\}.$$

6.3.2 Data-dependent regularization

In this section, we argue that under suitable conditions on the loss,

$$\phi_\gamma(\theta; (z, y)) = \ell(\theta; (z, y)) + \frac{1}{\gamma} \|\nabla_z \ell(\theta; (z, y))\|_2^2 + O\left(\frac{1}{\gamma^2} \right).$$

the robust surrogate loss (6.5) corresponds to a particular data-dependent regularization scheme. Let $\ell(\theta; (x, y))$ be the m-class softmax loss (6.2) given by

$$\ell(\theta; (x, y)) = -\log p_y(\theta, x) \quad \text{where} \quad p_j(\theta, x) := \frac{\exp(\theta_{c,j}^\top g(\theta, x))}{\sum_{l=1}^m \exp(\theta_{c,l}^\top g(\theta_f; x))}.$$

where $\theta_{c,j} \in \mathbb{R}^p$ is the j-th row of the classification layer weight $\theta_c \in \mathbb{R}^{p \times m}$. Then, the robust surrogate ϕ_γ is an approximate regularizer on the classification layer weights θ_c

$$\phi_\gamma(\theta, (x, y)) = \ell(\theta, (x, y)) + \frac{1}{\gamma} \left\| \theta_{c,y} - \sum_{j=1}^{m} p_j(\theta, x) \theta_{c,j} \right\|_2^2 + O\left(\frac{1}{\gamma^2}\right). \tag{6.11}$$

The expansion (6.11) shows that the robust surrogate (6.5) is roughly equivalent to data-dependent regularization where we minimize the distance between $\sum_{j=1}^{m} p_j(\theta, x)\theta_{c,j}$, our "average estimated linear classifier", to $\theta_{c,y}$, the linear classifier corresponding to the true label y. Letting $L(\theta) := 2 \max_{1 \le j \le m} \|\theta_{c,j}\|_2 \sum_{j=1}^{m} \|\theta_{c,j}\|_2$ for a fixed $\theta \in \Theta$, we have the following result whose proof we defer to Appendix D.

Theorem 2. *If* $Im(g(\theta_f; \cdot)) = \mathbb{R}^p$ *and* $\gamma > L(\theta)$*, the softmax loss* (6.2) *satisfies*

$$\frac{1}{\gamma + L} \left\| \theta_{c,y} - \sum_{j=1}^{m} p_j(\theta, x)\theta_{c,j} \right\|_2^2 \le \phi_\gamma(\theta, (x, y)) - \ell(\theta, (x, y)) \le \frac{1}{\gamma - L} \left\| \theta_{c,y} - \sum_{j=1}^{m} p_j(\theta, x)\theta_{c,j} \right\|_2^2.$$

6.4 Experiments

We evaluate our method for both classification and semantic segmentation settings, following the evaluation scenarios of domain adaptation techniques [39, 153, 62], though in our case the target domains are unknown at training time. We summarize our experimental setup including implementation details, evaluation metrics and datasets for each task. We compare our method against the Empirical Risk Minimization (ERM) baseline in all of our results.

Digit classification. We train on MNIST [82] dataset and test on MNIST-M [39], SVHN [111], SYN [39] and USPS [28]. We use $10,000$ digit samples

for training and evaluate our models on the respective test sets of the different target domains, using accuracy as a metric. In order to work with comparable datasets, we resized all the images to 32×32, and treated images from MNIST and USPS as RGB. We use a CNN [80] with architecture *conv-pool-conv-pool-fc-fc-softmax* and set the hyperparameters $\alpha = 0.0001$, $\eta = 1.0$, $T_{\min} = 100$ and $T_{\max} = 15$. In the minimization phase, we use Adam [75] with batch size equal to 32[1].

Semantic scene segmentation. We use the SYTHIA[124] dataset for semantic segmentation. The dataset contains images from different locations (we use *Highway*, *New York-like City* and *Old European Town*), and different weather/time/date conditions (we use *Dawn*, *Fog*, *Night*, *Spring* and *Winter*. We train models on a source domain and test on other domains, using the standard mean Intersection Over Union *(mIoU)* metric to evaluate our performance [33]. We arbitrarily chose images from the left front camera throughout our experiments. For each one, we sample 900 random images (resized to 192×320 pixels) from the training set. We use a Fully Convolutional Network (FCN) [91], with a ResNet-50 [55] body and set the hyperparameters $\alpha = 0.0001$, $\eta = 2.0$, $T_{\min} = 500$ and $T_{\max} = 50$. For the minimization phase, we use Adam [75] with batch size equal to 8.

6.4.1 Results on digit classification

In this section, we present and discuss the results on the digit classification experiment. The baselines (accuracies achieved by models trained with ERM) are:

[1]Models were implemented using Tensorflow, and training procedures were performed on NVIDIA GPUs. Code: `https://github.com/ricvolpi/generalize-unseen-domains`

- SVHN: 0.283 ± 0.032
- MNIST-M: 0.548 ± 0.021
- SYN: 0.406 ± 0.022
- USPS: 0.789 ± 0.017

Firstly, we are interested in analyzing the role of the semantic constraint we impose. Figure 6.1 *(top)* shows performances associated with models trained with Algorithm 3 with $K = 1$ and $\gamma = 10^4$, with the constraint in the semantic space (as discussed in Section 6.2) and in the pixel space [139], in blue and yellow, respectively. Figure 6.1 *(bottom)* shows performances of models trained with our method using different values of the hyperparameter γ (with $K = 2$) and with ERM (*blue* and *red* bars, respectively). These plots show (i) that moving the constraint at the semantic level carries benefits when models are tested on unseen domains and (ii) that models trained with Algorithm 3 outperform models train with ERM (*red* lines) for any value of γ on out-of-sample domains (SVHN, MNIST-M and SYN). The latter result is a rather desired achievement, since this hyperparameter cannot be properly cross-validated. On USPS, our method causes accuracy to drop since MNIST and USPS are very similar datasets, thus the image domain that USPS belongs to is not explored by our algorithm during the training procedure, which optimizes for worst case performance.

Figure 6.2 *(top)* reports results related to models trained with our method (*blue* bars), varying the number of iterations K and fixing $\gamma = 1.0$, and results related to ERM (*red* bars) and Dropout [141] (*yellow* bars). Also in this experiments, it can be observed that our method leads to statistically significant, improved performances when the models are tested on SVHN,

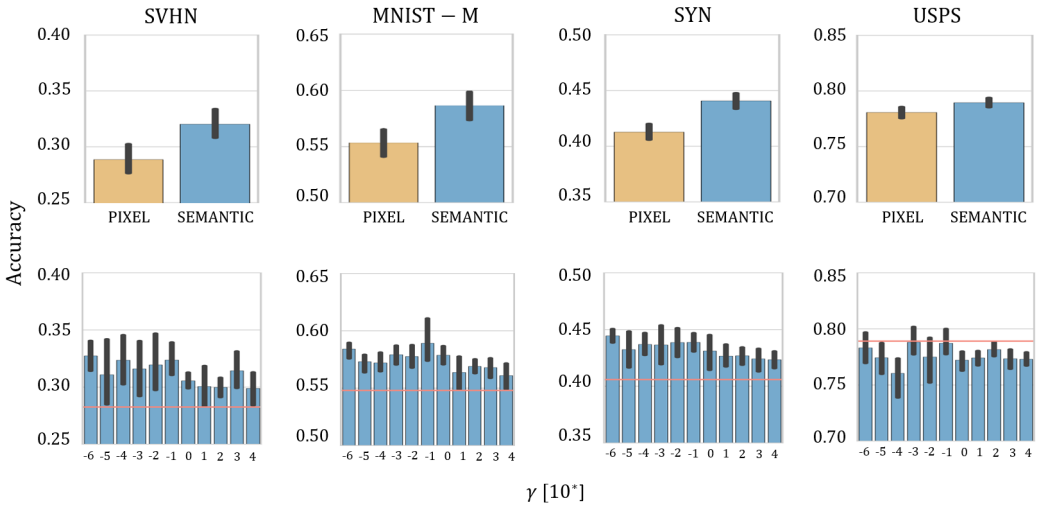


Figure 6.1. Results associated with models trained with $10,000$ MNIST samples and tested on SVHN, MNIST-M, SYN and USPS (1^{st}, 2^{nd}, 3^{rd} and 4^{th} columns, respectively). *Top:* comparison between distances in the pixel space (yellow) and in the semantic space (blue), with $K = 1$ and $\gamma = 10^4$. *Bottom:* comparison between our method with $K = 2$ and different γ values (blue bars) and ERM (red line). *Black* bars indicate the range of accuracies spanned. The reported results are obtained by averaging over 10 different runs.

MNIST-M and SYN, outperforming both ERM and Dropout [141]. In Figure 6.2 *(middle)*, we compare models trained with ridge regularization (*green* bars) with models trained with Algorithm 3 (with $K = 1$ and $\gamma = 1.0$) and ridge regularization; these results show that our method can potentially benefit from other regularization approaches, as in this case we observed that the two effects sum up. Table 6.4.1 reports numerical results obtained running our algorithm with different values for γ and K.

Finally, we report the results obtained by learning an ensemble of models. Since the hyperparameter γ is nontrivial to set a priori, we use the softmax confidences (6.9) to choose which model to use at test time. We learn ensemble of models, each of which is trained by running Algorithm 3 with different values of the γ as $\gamma = 10^{-i}$, with $i = \{0, 1, 2, 3, 4, 5, 6\}$. Figure 6.2 *(bottom)* shows the comparison between our method with different numbers of iterations

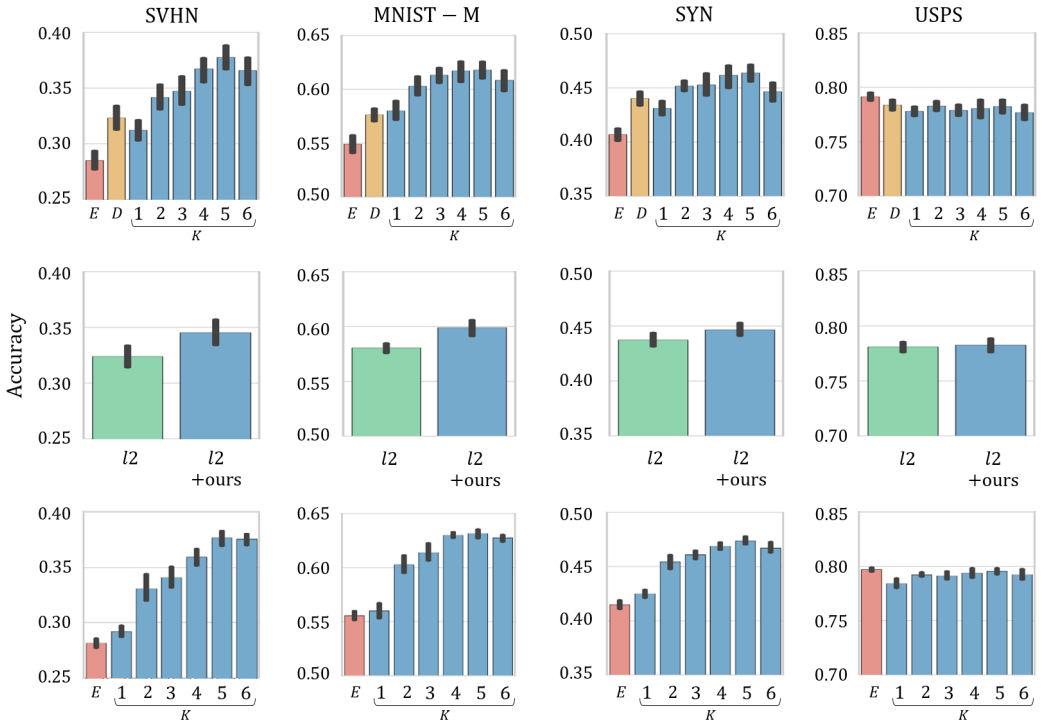


Figure 6.2. Results associated with models trained with 10,000 MNIST samples and tested on SVHN, MNIST-M, SYN and USPS (1^{st}, 2^{nd}, 3^{rd} and 4^{th} columns, respectively). *Top:* comparison between our method with $\gamma = 1.0$ and different number of iterations K, ERM and Dropout [141]. *Middle:* results obtained with the ensemble method by varying K. *Bottom:* comparison between models regularized with ridge (green) and with ridge + our method with K = 1 (blue). *Black* bars indicate the range of accuracies spanned. The reported results are obtained by averaging over 10 different runs.

K (blue) and ERM *(red)*. In order to separate the role of ensemble learning, we learn an ensemble of baseline models each corresponding to a different initialization. We fix the number of models in the ensemble to be the same for both the baseline (ERM) and our method. As it can be observed, by comparing Figure 6.2 *(bottom)* with Figure 6.2 *(top)* and Figure 6.1 *(bottom)*, our ensemble approach yields higher accuracy in different testing scenarios, more substantially for larger number of iterations K. In particular, it is worth noting the reduced performance gap detectable in the USPS experiment.

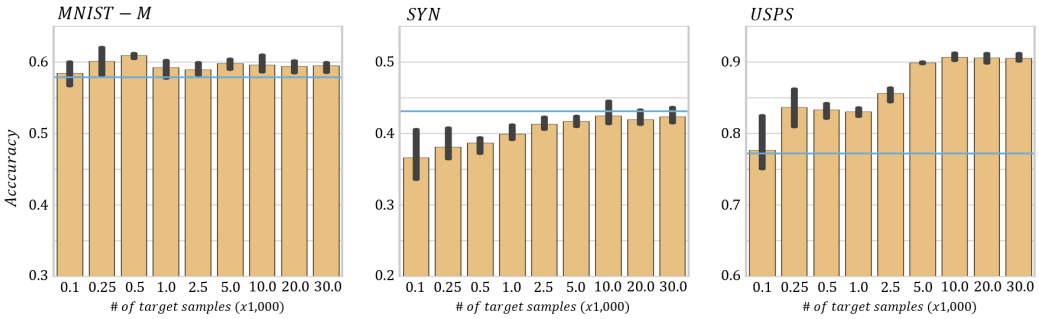


Figure 6.3. Results obtained by running ADDA algorithm [153] using $10,000$ labeled MNIST samples and a number of target samples indicated on the x-axis. The *blue* lines indicate results obtained with our method with $K = 2$ and $\gamma = 1.0$. Test sets are MNIST-M *(left)*, SYN *(middle)* and USPS *(right)*.

Comparison with domain adaptation. Figure 6.3 reports a comparison between our method *(blue)* and the unsupervised domain adaptation algorithm ADDA [153] *(yellow)*, by varying the number of target images fed to the latter during training. Note that, since unsupervised domain adaptation algorithms make use of target data during training while our method does not, the comparison is not fair. However, we are interested in evaluating to which extent our method can compete with a well performing unsupervised domain adaptation algorithm [153]. While on MNIST $\rightarrow$ USPS split ADDA clearly outperforms our method, on MNIST $\rightarrow$ MNIST-M the accuracies reached by our method are just slightly lower than the ones reached by ADDA, and on MNIST $\rightarrow$ SYN our method outperforms it, even if the domain adaptation algorithm has access to a large number of samples from the target domain. Finally, note that MNIST $\rightarrow$ SVHN results are not provided because ADDA would not converge on this split (in effect, these results are neither reported in the original work [153]). Instead, models trained on MNIST samples using our method better generalize to SVHN, as shown in Figure 6.3.

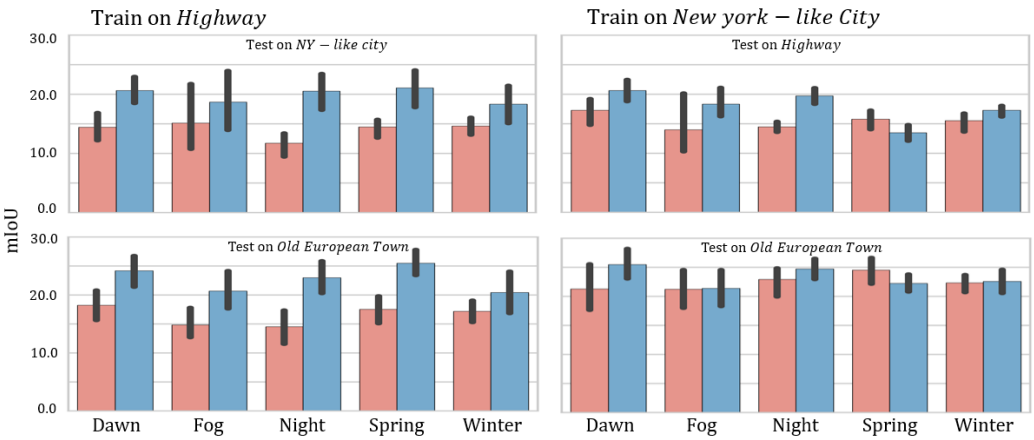


Figure 6.4. Results obtained with semantic segmentation models trained with ERM *(red)* and our method with $K = 1$ and $\gamma = 1.0$ *(blue)*. *Leftmost* panels are associated with models trained on *Highway*, *rightmost* panels are associated with models trained on *New York-like City*. Test datasets are *Highway*, *New York-like City* and *Old European Town*.

6.4.2 Results on semantic scene segmentation

We report a comparison between models trained with ERM and models trained with our method (Algorithm 3 with $K = 1$). We set $\gamma = 1.0$ in every experiment, but let us stress that this is an arbitrary value, we did not observe a strong correlation between the different values and the general behavior of the models in this case. Its role would be much more meaningful in an ensemble setting, where each model would be associated with a different level of robustness, as discussed in Section 6.2. In this setting, we do not apply the ensemble approach, but only evaluate the performances of the single models. The main reason for this choice is the fact that the heuristics developed to choose the correct model at test time in effect cannot be applied in a straightforward fashion to a semantic segmentation problem. One could apply it to the single pixels and combine the outputs of the different models according to the softmax distributions. Some preliminary results showed that this approach could be applicable, but we reserve a detailed analysis of it for

future works.

Figure 6.4 reports results obtained. Specifically, *leftmost* plots report results associated with models trained on sequences from the *Highway* split and tested on the *New York-like City* and the *Old European Town* splits (*top-left* and *bottom-left*, respectively); *rightmost* plots report results associated with models trained on sequences from the *New York-like City* split and tested on the *Highway* and the *Old European Town* splits (*top-right* and *bottom-right*, respectively). The training sequences (*Dawn*, *Fog*, *Night*, *Spring* and *Winter*) are indicate on the x-axis. *Red* and *blue* bars indicate average mIoUs achieved by models trained with ERM and by models trained with our method, respectively. These results were calculated by averaging over the mIoUs obtained with each model on the different conditions of the test set. As can be observed, models trained with our method mostly better generalize to unknown data distributions. In particular, our method always outperforms the baseline by a statistically significant margin when the training images are from *Night* scenarios. This is since the baseline model trained on images from *Night* is strongly biased towards dark scenery, while, as a consequence of training over worst-case distributions, our models can overcome this strong bias and better generalize across different unseen domains. Table 6.4.2 reports numerical results obtained.

6.5 Conclusions and future work

We study a new adversarial data augmentation procedure that learns to better generalize across unseen data distributions, and define an ensemble method to exploit this technique in a classification framework. This is in contrast to domain adaptation algorithms, which require a sufficient number of samples

from a known, a priori fixed target distribution. Our experimental results show that our iterative procedure provides broad generalization behavior on digit recognition and cross-season and cross-weather semantic segmentation tasks.

For future work, we hope to extend the ensemble methods by defining novel decision rules. The proposed heuristics (6.9) only apply to classification settings, and extending them to a broad realm of tasks including semantic segmentation is an important direction. Many theoretical questions still remain. For instance, quantifying the behavior of data-dependent regularization schemes presented in Section 6.3 would help us better understand adversarial training methods in general.

Table 6.1. Results obtained by training models with Algorithm 1 on $10,000$ MNIST samples and testing them on SVHN, MNIST-M, SYN and USPS. Results are averaged over 20 different runs.

	K=1	K=2	K=3	K=4
SVHN				
$\gamma = 10^{-5}$	0.284 ± 0.036	0.311 ± 0.033	0.316 ± 0.036	0.331 ± 0.026
$\gamma = 10^{-4}$	0.331 ± 0.018	0.324 ± 0.026	0.336 ± 0.020	0.325 ± 0.030
$\gamma = 10^{-3}$	0.294 ± 0.023	0.316 ± 0.029	0.309 ± 0.024	0.343 ± 0.017
$\gamma = 10^{-2}$	0.290 ± 0.041	0.320 ± 0.030	0.341 ± 0.030	0.346 ± 0.033
$\gamma = 10^{-1}$	0.284 ± 0.007	0.324 ± 0.017	0.307 ± 0.026	0.323 ± 0.029
$\gamma = 10^{0}$	0.284 ± 0.012	0.306 ± 0.008	0.314 ± 0.022	0.335 ± 0.029
$\gamma = 10^{1}$	0.305 ± 0.031	0.301 ± 0.035	0.316 ± 0.027	0.343 ± 0.030
$\gamma = 10^{2}$	0.304 ± 0.032	0.300 ± 0.017	0.327 ± 0.026	0.321 ± 0.034
$\gamma = 10^{3}$	0.289 ± 0.030	0.314 ± 0.032	0.300 ± 0.017	0.304 ± 0.025
$\gamma = 10^{4}$	0.300 ± 0.020	0.299 ± 0.028	0.325 ± 0.015	0.340 ± 0.026
MNIST-M				
$\gamma = 10^{-5}$	0.564 ± 0.024	0.573 ± 0.010	0.573 ± 0.024	0.589 ± 0.017
$\gamma = 10^{-4}$	0.583 ± 0.011	0.572 ± 0.010	0.586 ± 0.015	0.578 ± 0.031
$\gamma = 10^{-3}$	0.562 ± 0.026	0.579 ± 0.010	0.567 ± 0.023	0.601 ± 0.018
$\gamma = 10^{-2}$	0.539 ± 0.037	0.578 ± 0.013	0.590 ± 0.014	0.598 ± 0.014
$\gamma = 10^{-1}$	0.556 ± 0.017	0.589 ± 0.021	0.576 ± 0.018	0.576 ± 0.019
$\gamma = 10^{0}$	0.557 ± 0.017	0.579 ± 0.009	0.571 ± 0.010	0.584 ± 0.024
$\gamma = 10^{1}$	0.568 ± 0.022	0.564 ± 0.028	0.579 ± 0.024	0.589 ± 0.016
$\gamma = 10^{2}$	0.564 ± 0.025	0.569 ± 0.013	0.579 ± 0.019	0.578 ± 0.021
$\gamma = 10^{3}$	0.558 ± 0.016	0.568 ± 0.017	0.568 ± 0.010	0.567 ± 0.021
$\gamma = 10^{4}$	0.567 ± 0.022	0.561 ± 0.023	0.570 ± 0.015	0.579 ± 0.016

Table 6.2. Results obtained by training models with Algorithm 1 on $10,000$ MNIST samples and testing them on SVHN, MNIST-M, SYN and USPS. Results are averaged over 20 different runs.

	K=1	K=2	K=3	K=4
SYN				
$\gamma = 10^{-5}$	0.409 ± 0.029	0.432 ± 0.020	0.437 ± 0.024	0.443 ± 0.014
$\gamma = 10^{-4}$	0.439 ± 0.011	0.437 ± 0.011	0.446 ± 0.018	0.440 ± 0.022
$\gamma = 10^{-3}$	0.417 ± 0.018	0.437 ± 0.021	0.436 ± 0.017	0.450 ± 0.010
$\gamma = 10^{-2}$	0.417 ± 0.022	0.439 ± 0.015	0.447 ± 0.020	0.450 ± 0.014
$\gamma = 10^{-1}$	0.405 ± 0.011	0.439 ± 0.009	0.438 ± 0.018	0.439 ± 0.021
$\gamma = 10^{0}$	0.418 ± 0.004	0.431 ± 0.017	0.426 ± 0.021	0.441 ± 0.013
$\gamma = 10^{1}$	0.421 ± 0.016	0.427 ± 0.020	0.436 ± 0.020	0.445 ± 0.016
$\gamma = 10^{2}$	0.427 ± 0.017	0.427 ± 0.016	0.436 ± 0.021	0.432 ± 0.014
$\gamma = 10^{3}$	0.410 ± 0.027	0.424 ± 0.019	0.422 ± 0.019	0.418 ± 0.015
$\gamma = 10^{4}$	0.422 ± 0.018	0.423 ± 0.015	0.441 ± 0.010	0.443 ± 0.016
USPS				
$\gamma = 10^{-5}$	0.775 ± 0.016	0.774 ± 0.017	0.778 ± 0.010	0.782 ± 0.016
$\gamma = 10^{-4}$	0.781 ± 0.010	0.760 ± 0.021	0.772 ± 0.013	0.774 ± 0.021
$\gamma = 10^{-3}$	0.758 ± 0.012	0.788 ± 0.014	0.771 ± 0.011	0.784 ± 0.011
$\gamma = 10^{-2}$	0.765 ± 0.012	0.775 ± 0.024	0.772 ± 0.021	0.775 ± 0.011
$\gamma = 10^{-1}$	0.773 ± 0.011	0.787 ± 0.013	0.774 ± 0.011	0.776 ± 0.018
$\gamma = 10^{0}$	0.778 ± 0.007	0.772 ± 0.010	0.774 ± 0.017	0.768 ± 0.021
$\gamma = 10^{1}$	0.767 ± 0.018	0.774 ± 0.013	0.779 ± 0.016	0.773 ± 0.014
$\gamma = 10^{2}$	0.774 ± 0.014	0.782 ± 0.013	0.776 ± 0.018	0.771 ± 0.021
$\gamma = 10^{3}$	0.774 ± 0.013	0.774 ± 0.017	0.775 ± 0.012	0.763 ± 0.025
$\gamma = 10^{4}$	0.778 ± 0.013	0.773 ± 0.012	0.774 ± 0.012	0.781 ± 0.011

Table 6.3. Results *(mIoUs)* associated with the experiments on SYNTHIA dataset. The *first* column indicate the training set. The *second* column indicate the method used: Empirical Risk Minimization *(ERM)* and our method *(Ours)* with $K = 1$ and $\gamma = 1.0$. Remaining columns indicate the test set.

		New York-like City					Old European Town				
		Dawn	Fog	Night	Spring	Winter	Dawn	Fog	Night	Spring	Winter
	ERM	18.9	14.7	10.7	14.5	13.4	22.0	20.8	14.5	18.6	15.3
Highway/Dawn	*Ours*	**24.0**	**17.0**	**19.1**	**22.9**	**20.2**	**27.6**	**25.0**	**22.4**	**27.1**	**19.0**
	ERM	12.6	27.8	9.0	12.9	13.4	13.6	20.7	12.1	15.1	12.7
Highway/Fog	*Ours*	**17.4**	**28.4**	**11.0**	**18.4**	18.4	**18.5**	**27.5**	**16.4**	**22.0**	**19.0**
	ERM	13.0	7.7	13.9	13.2	10.9	16.6	11.5	19.0	15.7	9.9
Highway/Night	*Ours*	**18.5**	**14.5**	24.8	**22.9**	**22.0**	**22.2**	**20.1**	**28.1**	**25.5**	**19.1**
	ERM	15.2	16.0	10.8	15.8	14.8	18.8	21.2	14.7	19.2	13.9
Highway/Spring	*Ours*	**22.6**	**19.4**	14.6	**25.5**	**23.5**	**25.1**	**26.5**	**21.5**	**29.9**	**24.5**
	ERM	14.1	15.9	11.7	14.8	16.8	15.2	19.3	14.6	16.9	20.0
Highway/Winter	*Ours*	**16.9**	**17.4**	**12.5**	**21.0**	**24.0**	**17.0**	**20.5**	**14.9**	**23.1**	**26.8**
		Highway					Old European Town				
		Dawn	Fog	Night	Spring	Winter	Dawn	Fog	Night	Spring	Winter
	ERM	19.6	19.1	13.1	18.8	15.9	27.9	23.5	16.3	21.7	17.0
NY.Like C./ Dawn	*Ours*	**22.8**	**22.8**	**17.8**	**21.4**	**18.5**	**31.0**	**25.9**	**22.4**	**26.0**	22.3
	ERM	12.5	15.9	9.1	11.8	10.7	24.2	26.5	17.8	21.7	16.0
NY.Like C./Fog	*Ours*	**15.4**	**23.1**	**16.3**	**18.7**	**18.2**	**17.3**	**26.4**	**17.5**	**24.3**	21.6
	ERM	14.9	14.7	16.3	13.5	13.1	25.4	24.7	24.4	23.3	17.0
NY.Like C./Night	*Ours*	**19.4**	**20.2**	**22.1**	**19.7**	**17.3**	**23.3**	**23.9**	**27.2**	**27.2**	**22.1**
	ERM	**17.1**	**18.0**	**12.8**	**16.3**	**14.8**	**26.6**	**27.0**	**20.4**	**26.3**	**22.5**
NY.Like C./Spring	*Ours*	14.5	14.7	11.8	15.2	11.2	21.9	21.9	19.7	24.8	22.9
	ERM	16.1	17.3	11.9	16.5	16.0	**21.3**	**23.8**	19.4	24.1	23.2
NY.Like C./Winter	*Ours*	**18.1**	**18.2**	**15.2**	**17.8**	**17.3**	21.0	21.0	**19.9**	**25.5**	**25.6**

Chapter 7

Conclusions

In this thesis, the generalization and adaptation properties of neural networks are investigated. A series of different approaches are presented, to cope with generalization problematics at different levels.

First, a novel way to perform dropout training is introduced [105], combining Curriculum Learning [9] and standard Dropout [141]. This method not only shows better generalization properties, but also allows a series of theoretical interpretations.

Next, the problem of unsupervised domain adaptation is faced. Two new methods based on Generative Adversarial Networks [45] are presented, to bridge the domain gap between a source and a target distribution fixed a priori, showing performance competitive with the state of the art in several unsupervised domain adaptation benchmarks. Future directions for these works are constituted by the application of these approaches to more realistic and challenging problems.

Finally, the applicability of domain adaptation in realistic settings is questioned, and a novel problem is proposed. In the proposed setting, the goal is to generalize to unknown domains by relying only on data from a single

population source. A solution to this problem is proposed, based on a worst-case formulation over data distributions which are ρ-distant from the source one in a semantic space. For classification problems, an ensemble solution is also proposed. Results show the effectiveness of the approach on cross-dataset digit recognition and cross-season/weather semantic segmentation. Future work includes the definition of a more effective selection rule for the ensemble method, which could potentially be applied also to semantic segmentation problems.

Bibliography

[1] Anonymous. Large scale gan training for high fidelity natural image synthesis. In *Submitted to International Conference on Learning Representations*, 2019. under review.

[2] Shuang Ao, Xiang Li, and Charles Ling. Fast generalized distillation for semi-supervised domain adaptation, 2017.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[4] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). *CoRR*, abs/1703.00573, 2017.

[5] Sanjeev Arora, Andrej Risteski, and Yi Zhang. Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations*, 2018.

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[7] Justin Bayer, Christian Osendorfer, and Nutan Chen. On fast dropout and its applicability to recurrent networks. In *CoRR:1311.0701*, 2013.

[8] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 137–144. MIT Press, 2007.

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

[10] David Berthelot, Tom Schumm, and Luke Metz. BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717, 2017.

[11] Bharath Bhushan Damodaran, Benjamin Kellenberger, Remi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[12] Chris M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Comput.*, 7(1):108–116, 1995.

[13] Jose Blanchet and Karthyek Murthy. Quantifying distributional model risk via optimal transport. *arXiv:1604.01446 [math.PR]*, 2016.

[14] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the*

2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06, pages 120–128, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[15] J Frédéric Bonnans and Alexander Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

[16] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[17] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 343–351. Curran Associates, Inc., 2016.

[18] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

[19] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[20] Gulcehre Caglar, Sotelo Jose, Moczulski Marcin, and Yoshua Bengio. A robust adaptive stochastic gradient method for deep learning. In *CoRR:1703.00788*, 2017.

[21] Jacopo Cavazza and Vittorio Murino. Active Regression with Adaptive Huber loss. In *CoRR:1606.01568*, 2016.

[22] Xi Chen, Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2172–2180. Curran Associates, Inc., 2016.

[23] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains. In *in ICML Workshop on Challenges in Representation Learning*, 2013.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[25] Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *NIPS*. 2009.

[26] Hal Daumé, III, Abhishek Kumar, and Avishek Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, DANLP 2010, pages 53–59, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[28] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I. Guyon. Advances in neural information processing systems 1. chapter Neural Network Recognizer for Hand-written Zip Code Digits, pages 323–331. 1989.

[29] J. Donahue, J. Hoffman, E. Rodner, K. Saenko, and T. Darrell. Semi-supervised domain adaptation with instance constraints. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 668–675, June 2013.

[30] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[31] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[32] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *CoRR*, abs/1606.00704, 2016.

[33] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[34] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1), 2000.

[35] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR workshop*, 2004.

[36] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment.

In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 2960–2967, 2013.

[37] William Finnoff, Ferdinand Hergert, and Hans-Georg Zimmermann. Improving model selection by nonconvergent methods. *Neural Networks*, 6(6):771–783, 1993.

[38] Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In *International Conference on Learning Representations*, 2018.

[39] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1180–1189, 2015.

[40] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1), January 2016.

[41] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2551–2559, 2015.

[42] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *In Proceedings of the Twenty-eight International Conference on Machine Learning, ICML*, 2011.

[43] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *In CVPR*, 2012.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[45] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[46] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[47] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[48] Raghuraman Gopalan and Ruonan Li. Domain adaptation for object recognition: An unsupervised approach. In *In ICCV*, 2011.

[49] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, March 2012.

[50] G. Griffin, A. Holub, and P Perona. Caltech-256 object category dataset. In *Technical Report 7694, California Institute of Technology*, 2007.

[51] Saurabh Gupta, Ross Girshick, Pablo Arbel´ aez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and

segmentation. In *European Conference on Computer Vision (ECCV)*, 2014.

[52] Benjamin D. Haeffele and René Vidal. Global optimality in tensor factorization, deep learning, and beyond. In *CoRR:1506.07540*, 2015.

[53] Philip Haeusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. Associative domain adaptation. In *International Conference on Computer Vision (ICCV)*, 2017.

[54] LK. Hansen and CE. Rasmussen. Pruning from adaptive regularization. *Neural Computation*, 6(6):1222–1231, 1994.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[56] Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness. *arXiv preprint arXiv:1710.11469*, 2017.

[57] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, abs/1610.02136, 2016.

[58] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6626–6637. Curran Associates, Inc., 2017.

[59] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[60] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *CoRR:1207.0580*, 2012.

[61] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[62] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, abs/1612.02649, 2016.

[63] De-An Huang and Yu-Chiang Frank Wang. Coupled dictionary and feature space learning with applications to cross-domain image synthesis and recognition. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 2496–2503, 2013.

[64] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269, 2017.

[65] Dong hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 2013.

[66] Hal Daume III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[67] Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *CoRR*, abs/1109.6341, 2011.

[68] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[69] Ba Jimmy and Brendan Frey. Adaptive dropout for training deep neural networks. In *NIPS*, 2016.

[70] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[71] Meina Kan, Shiguang Shan, and Xilin Chen. Bi-shifting auto-encoder for unsupervised domain adaptation. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 3846–3854, 2015.

[72] Guoliang Kang, Liang Zheng, Yan Yan, and Yi Yang. Deep adversarial attention alignment for unsupervised domain adaptation: the benefit of target expectation maximization. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[73] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[74] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision (ECCV)*, Florence, Italy, October 2012.

[75] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[76] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[77] Naveen Kodali, Jacob D. Abernethy, James Hays, and Zsolt Kira. How to train your DRAGAN. *CoRR*, abs/1705.07215, 2017.

[78] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[80] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December 1989.

[81] Y. LeCun, B. Boser, JS Denker, D. Henderson, R. Howard, W. Hubbard, and LD Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, pages 541–551, 1989.

[82] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[83] Jaeho Lee and Maxim Raginsky. Minimax statistical learning and domain adaptation with wasserstein distances. *arXiv preprint arXiv:1705.07815*, 2017.

[84] K Levenberg. A method for the solution of certain problems in least squares. quart. appl. math. 2. 1944.

[85] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[86] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[87] Zhe Gong Li and Tianbao Boqing Yang. Improved dropout for shallow and deep learning. In *NIPS*, 2016.

[88] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5168–5177, 2017.

[89] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017.

[90] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 469–477. Curran Associates, Inc., 2016.

[91] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[92] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 97–105, 2015.

[93] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 97–105, 2015.

[94] David López-Paz, José Miguel Hernández-Lobato, and Bernhard Schölkopf. Semi-supervised domain adaptation with non-parametric copulas. In *NIPS*, pages 674–682, 2012.

[95] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? A large-scale study. *CoRR*, abs/1711.10337, 2017.

[96] M. Mancini, S. R. Bulò, B. Caputo, and E. Ricci. Robust place categorization with deep domain generalization. *IEEE Robotics and Automation Letters*, 3(3):2093–2100, July 2018.

[97] Massimiliano Mancini, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Best sources forward: Domain generalization through source-specific nets. In *2018 IEEE International Conference on Image Processing, ICIP 2018, Athens, Greece, October 7-10, 2018*, pages 1353–1357, 2018.

[98] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Boosting domain adaptation by discovering latent domains. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[99] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016.

[100] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[101] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[102] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[103] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[104] Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *International Conference on Learning Representations*, 2018.

[105] Pietro Morerio, Jacopo Cavazza, Riccardo Volpi, René Vidal, and Vittorio Murino. Curriculum dropout. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2017.

[106] N. Morgan and H. Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 630–637. Morgan-Kaufmann, 1990.

[107] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 10–18, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[108] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[109] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010.

[110] Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[111] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[112] Karl Pearson. *On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling*, pages 11–28. Springer New York, New York, NY, 1992.

[113] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *CoRR*, abs/1710.06924, 2017.

[114] Pedro O. Pinheiro. Unsupervised domain adaptation with similarity learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[115] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[116] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of*

Machine Learning Research, pages 2847–2854, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[117] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 779–788, 2016.

[118] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525, 2017.

[119] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[120] Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 217–225. Curran Associates, Inc., 2016.

[121] Steven J. Rennie, Vaibhava Goel, and Samuel Thomas. Annealed dropout training of deep networks. In *Proceedings onf the IEEE Workshop on SLT*, pages 159–164, 2014.

[122] Salah Rifai, Xavier Glorot, Bengio Yoshua, and Pascal Vincent. Adding noise to the input of a model trained with a regularized objective. In *CoRR:1104.3250*, 2011.

[123] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[124] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[125] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *CoRR*, abs/1603.06432, 2016.

[126] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[127] Paolo Russo, Fabio M. Carlucci, Tatiana Tommasi, and Barbara Caputo. From source to target and back: Symmetric bi-directional adaptive gan. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[128] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 213–226, Berlin, Heidelberg, 2010. Springer-Verlag.

[129] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *Proceedings of*

the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, pages 2988–2997, 2017.

[130] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial dropout regularization. In *International Conference on Learning Representations*, 2018.

[131] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[132] Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[133] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *In Advances In Neural Information Processing Systems*, page 2110–2118, 2016.

[134] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018.

[135] Sumit Shekhar, Vishal M. Patel, Hien Van Nguyen, and Rama Chellappa. Generalized domain-adaptive dictionaries. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 361–368, 2013.

[136] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A DIRT-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

[137] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, , and Rob Fergus. Indoor segmentation and support infer- ence from rgbd images. In *European Conference on Computer Vision (ECCV)*, 2012.

[138] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[139] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018.

[140] Mahdi Soltanolkotabi, Ehsan Elhamifar, and Emmanuel J. Candès. Robust subspace clustering. In *CoRR:1301.2603*, 2013.

[141] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1), January 2014.

[142] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[143] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised Learning of Video Representations using LSTMs. In *CoRR:1502.04681*, 2015.

[144] Masashi Sugiyama, Song Liu, Marthinus Christoffel du Plessis, Masao Yamanaka, Makoto Yamada, Taiji Suzuki, and Takafumi Kanamori.

Direct divergence approximation between probability distributions and its applications in machine learning. *JCSE*, 7(2):99–111, 2013.

[145] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, Oct 2012.

[146] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2058–2065, 2016.

[147] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. In *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, pages 443–450, 2016.

[148] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[149] Robert Tibshirani. Regression shrinkage and selection via the lasso. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 58:267–288, 1994.

[150] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed problems.* W.H. Winston, 1977.

[151] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transfer-

ring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017.

[152] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1521–1528, Washington, DC, USA, 2011. IEEE Computer Society.

[153] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[154] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[155] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[156] L.J.P van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[157] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[158] Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. Adversarial feature augmentation for unsupervised domain adaptation.

In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[159] Riccardo Volpi*, Hongseok Namkoong*, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advanced in Neural Information Processing Systems (NIPS) 32*, December 2018.

[160] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 613–621. Curran Associates, Inc., 2016.

[161] Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *NIPS*. 2013.

[162] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[163] S. Wang and C. D. Manning. Fast dropout training. In *ICML*, pages 118–126, 2013.

[164] Haibing Wu and Xiaodong Gu. Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1–10, 2015.

[165] Zheng Xu, Wen Li, Li Niu, and Dong Xu. Exploiting low-rank structure from latent domains for domain generalization. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, pages 628–643, 2014.

[166] Ting Yao, Yingwei Pan, Chong-Wah Ngo, Houqiang Li, and Tao Mei. Semi-supervised domain adaptation with subspace learning for visual recognition. In *CVPR*, pages 2142–2150. IEEE Computer Society, 2015.

[167] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[168] Shangfei Zhai and Zhongfei Mark Zhang. Dropout training of matrix factorization and autoencoders for link prediction in sparse graphs. In *CoRR:1512.04483*, 2015.

[169] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NIPS*. 2014.

[170] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

Appendix A

Supplementary Material for Chapter 3

A.1 Theoretical proofs

In this Section, we provide all theoretical proofs for the statements reported in Chapter 3[1]. Section A.1.1 draws additional connections with regularization theory, demonstrating that Curriculum Dropout implements an adaptive regularization scheme. Eventually, in Section A.1.2, we formally prove that our dropout strategy is naturally interpretable within the curriculum learning paradigm [9], as we assert in Section 3.4.

A.1.1 Adaptive Regularization

In our work, we posit that our curriculum dropout can be interpreted as a smooth manner of imposing regularization. Precisely, the increase rate of neurons suppressions act as a progressive rule to simplify the overall model,

[1]Collaboration with Jacopo Cavazza, co-author of the paper *Curriculum Dropout* [105].

as to prevent overfitting. In this Section, we establish connections between curriculum dropout and regularization theory [34] with adaptive schemes [54, 25, 18, 140, 21]. In order to do so, consider a supervised regression or classification problem where the data $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$ are paired with corresponding labels $y_1, \dots, y_N \in \mathbb{R}$. Assume a least square fitting

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{N} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 = \min_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \tag{A.1}$$

to fit a linear model to explain the data, being $\mathbf{y} = [y_1, \dots, y_N]^\top$ and $\mathbf{X}$ the $N \times d$ matrix, whose i-th row $[X_{i1}, \dots, X_{id}] = \mathbf{x}_i^\top$. In the following pages, we will care of providing theoretical results which guarantee that dropout regularization on the un-regularized L_2 loss rewrites as a modification of the original loss, by adding a (data-dependent) regularizing term. More precisely, we apply the dropout formulation provided by [60, 142] to a classical least squares data fitting. As the following result show, once dropout is applied on an unregularized L^2 minimization, the latter problem rewrites as adding to the same loss a data-dependent regularization term which is modulated by $\theta(1-\theta)$.

Theorem 1 (Dropout - least squares)**.** *According to [60, 142], the dropout problem on the least squares fitting* (A.1)*, rewrites*

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathbb{E}_{\mathbf{r}} \left[\sum_{i=1}^{N} (y_i - \mathbf{w}^\top (\mathbf{r} \odot \mathbf{x}_i))^2 \right] = \min_{\mathbf{w}\in\mathbb{R}^d} \mathbb{E}_{\mathbf{r}} \|\mathbf{y} - \mathbf{X}\mathrm{diag}(\mathbf{r})\mathbf{w}\|_2^2, \tag{A.2}$$

being $\mathbf{r} = [r_1, \dots, r_d]$ *and* $r_j \sim \text{Bernoulli}(\theta)$ *i.i.d. It results*

$$\mathbb{E}_{\mathbf{r}} \|\mathbf{y} - \mathbf{X}\mathrm{diag}(\mathbf{r})\mathbf{w}\|_2^2 = \theta(1-\theta)\|\mathrm{diag}(\mathbf{X}^\top\mathbf{X})^{1/2}\mathbf{w}\|_2^2 + \|\mathbf{y} - \theta\mathbf{X}\mathbf{w}\|_2^2 \tag{A.3}$$

$$= \theta(1-\theta)\|\mathbf{w}\|^2_{\mathrm{diag}(\mathbf{X}^\top\mathbf{X})} + \|\mathbf{y} - \theta\mathbf{X}\mathbf{w}\|_2^2 \tag{A.4}$$

Proof Using the definition of the Euclidean norm and the linearity of the

expected value, we get

$$\mathbb{E}_{\mathbf{r}}\|\mathbf{y}-\mathbf{X}\text{diag}(\mathbf{r})\mathbf{w}\|_2^2 = \mathbb{E}_{\mathbf{r}}\left[\sum_{i=1}^{N}(y_i - \mathbf{w}^\top(\mathbf{r}\odot\mathbf{x}_i))^2\right] = \sum_{i=1}^{N}\mathbb{E}_{\mathbf{r}}\left[(y_i - \mathbf{w}^\top(\mathbf{r}\odot\mathbf{x}_i))^2\right]. \tag{A.5}$$

Apply the bias-variance decomposition $\mathbb{E}[Z^2] = \mathbb{V}[Z] + \mathbb{E}[Z]^2$, holding for any scalar random variable Z.

$$\mathbb{E}_{\mathbf{r}}\|\mathbf{y} - \mathbf{X}\text{diag}(\mathbf{r})\mathbf{w}\|_2^2 = \sum_{i=1}^{N}\left[\mathbb{V}_{\mathbf{r}}\left[y_i - \mathbf{w}^\top(\mathbf{r}\odot\mathbf{x}_i)\right] + \left(\mathbb{E}_{\mathbf{r}}\left[y_i - \mathbf{w}^\top(\mathbf{r}\odot\mathbf{x}_i)\right]\right)^2\right]. \tag{A.6}$$

The operator $\mathbb{V}_{\mathbf{r}}$ is invariant to deterministic translations. Therefore,

$$\mathbb{E}_{\mathbf{r}}\|\mathbf{y} - \mathbf{X}\text{diag}(\mathbf{r})\mathbf{w}\|_2^2 = \sum_{i=1}^{N}\left[\mathbb{V}_{\mathbf{r}}\left[-\mathbf{w}^\top(\mathbf{r}\odot\mathbf{x}_i)\right] + \left(\mathbb{E}_{\mathbf{r}}\left[y_i - \mathbf{w}^\top(\mathbf{r}\odot\mathbf{x}_i)\right]\right)^2\right]. \tag{A.7}$$

Once expanded the product $\mathbf{w}^\top(\mathbf{r}\odot\mathbf{x}_i)$ in components, we get

$$\mathbb{E}_{\mathbf{r}}\|\mathbf{y} - \mathbf{X}\text{diag}(\mathbf{r})\mathbf{w}\|_2^2 = \sum_{i=1}^{N}\left[\mathbb{V}_{\mathbf{r}}\left[-\sum_{j=1}^{d}w_j r_j X_{ij}\right] + \left(\mathbb{E}_{\mathbf{r}}\left[y_i - \sum_{j=1}^{d}w_j r_j X_{ij}\right]\right)^2\right] \tag{A.8}$$

For each i-th term of the summation, use the properties of variance and expected values with respect to linear combinations of independent random variables. This yields

$$\mathbb{E}_{\mathbf{r}}\|\mathbf{y} - \mathbf{X}\text{diag}(\mathbf{r})\mathbf{w}\|_2^2 = \sum_{i=1}^{N}\left[\sum_{j=1}^{d}w_j^2 X_{ij}^2\mathbb{V}_{\mathbf{r}}\left[r_j\right] + \left(y_i - \sum_{j=1}^{d}w_j X_{ij}\mathbb{E}_{\mathbf{r}}\left[r_j\right]\right)^2\right] \tag{A.9}$$

$$= \sum_{i=1}^{N}\left[\sum_{j=1}^{d}w_j^2 X_{ij}^2\cdot\theta(1-\theta) + \left(y_i - \sum_{j=1}^{d}w_j X_{ij}\cdot\theta\right)^2\right], \tag{A.10}$$

being the latter equality a direct consequence of the formulæ for the variance and the expected value for a Bernoulli(θ) distribution. Therefore, by highlighting the terms $\theta(1-\theta)$ and θ in front of the relative summations, we get

$$\mathbb{E}_{\mathbf{r}}\|\mathbf{y}-\mathbf{X}\text{diag}(\mathbf{r})\mathbf{w}\|_2^2 = \theta(1-\theta)\sum_{i=1}^{N}\sum_{j=1}^{d} w_j^2 X_{ij}^2 + \sum_{i=1}^{N}\left(y_i - \theta\sum_{j=1}^{d} X_{ij}w_j\right)^2. \tag{A.11}$$

By using the definition of Euclidean norm,

$$\mathbb{E}_{\mathbf{r}}\|\mathbf{y}-\mathbf{X}\text{diag}(\mathbf{r})\mathbf{w}\|_2^2 = \theta(1-\theta)\sum_{i=1}^{N}\sum_{j=1}^{d} w_j^2 X_{ij}^2 + \|\mathbf{y}-\theta\mathbf{X}\mathbf{w}\|_2^2 \tag{A.12}$$

Let us consider the first addend of (A.12) separately. By rearranging the summing ordering and replacing X_{ij}^2 with two identical copies of X_{ij}, we get

$$\sum_{i=1}^{N}\sum_{j=1}^{d} w_j^2 X_{ij}^2 = \sum_{j=1}^{d} w_j^2\left(\sum_{i=1}^{N} X_{ij}^2\right) = \sum_{j=1}^{d} w_j^2\left(\sum_{i=1}^{N} X_{ij}X_{ij}\right) \tag{A.13}$$

$$= \sum_{j=1}^{d} w_j^2\left(\sum_{i=1}^{N}(\mathbf{X})_{ji}^{\top}X_{ij}\right) = \sum_{j=1}^{d} w_j^2[\text{diag}(\mathbf{X}^{\top}\mathbf{X})]_{jj} \tag{A.14}$$

where we have exploited the transposition and the row-by-column product definitions. By squaring and square-rooting the second factor in the summation we obtain

$$\sum_{i=1}^{N}\sum_{j=1}^{d} w_j^2 X_{ij}^2 = \sum_{j=1}^{d} w_j^2([\text{diag}(\mathbf{X}^{\top}\mathbf{X})]_{jj}^{1/2})^2. \tag{A.15}$$

By noticing that the square-root of a diagonal matrix is a diagonal matrix whose entries are the square roots of the original entries, we obtain

$$\sum_{i=1}^{N}\sum_{j=1}^{d} w_j^2 X_{ij}^2 = \sum_{j=1}^{d} w_j^2([\text{diag}(\mathbf{X}^{\top}\mathbf{X})^{1/2}]_{jj})^2 = \sum_{j=1}^{d}(w_j[\text{diag}(\mathbf{X}^{\top}\mathbf{X})^{1/2}]_{jj})^2. \tag{A.16}$$

Apply the definition of row-by-column matrix product between a diagonal matrix and a vector.

$$\sum_{i=1}^{N}\sum_{j=1}^{d} w_j^2 X_{ij}^2 = \sum_{j=1}^{d}([\text{diag}(\mathbf{X}^\top\mathbf{X})^{1/2}\mathbf{w}]_j)^2 \tag{A.17}$$

$$= \|\text{diag}(\mathbf{X}^\top\mathbf{X})^{1/2}\mathbf{w}\|_F^2, \tag{A.18}$$

where, in (A.18), we used the definition of Frobenius norm. Replacing (A.18) in (A.12), leads to to prove (A.3).

In order to elicit (A.4), it is enough to notice that (A.15) can be rewritten as

$$\sum_{j=1}^{d} w_j^2[\text{diag}(\mathbf{X}^\top\mathbf{X})]_{jj} = \sum_{j=1}^{d} w_j[\text{diag}(\mathbf{X}^\top\mathbf{X})]_{jj}w_j = \mathbf{w}^\top\text{diag}(\mathbf{X}^\top\mathbf{X})\mathbf{w}, \tag{A.19}$$

being the last term equivalent to $\|\mathbf{w}\|_{\text{diag}(\mathbf{X}^\top\mathbf{X})}$, by exploiting the definition of norm induced by a symmetric and positive definite matrix [**?**]. Therefore, (A.19), once plugged into (A.15), leads to prove (A.4). This completes the proof. □

We can apply the identical analysis to the case of a deep neural network. In such a case, the input data matrix $\mathbf{X}$ is processed across subsequences of ℓ linear layers (represented by weights $\mathbf{W}^{(\ell)}$), with intermediate gating functions, pooling and feature normalization steps. Despite the latter non-linearities, since the values sampled from a Bernoulli are always either 0 or 1, it is enough to enumerate all the possible binary combinations of activations/inhibitions, accounting for the probability of their occurrence. This allows to retrieve (a different regularization term but) the same weighting factor $\theta(1-\theta)$.

Finally, let us note that, despite in the previous analysis the parameter θ was considered to be fixed, we can easily generalize it for $\theta = \theta(t)$. Precisely,

it is enough to fix t arbitrary, apply the previous proofs by replacing θ with $\theta(t)$ and finally exploit the generality of t.

This concludes the theoretical analysis reported in the paper.

A.1.2 Curriculum Dropout, Curriculum Learning

In this Section, we justify the name Curriculum Dropout by proving its equivalence to Curriculum Learning [9]. Precisely, with respect to the definition of a curriculum distribution provided in [9], we show that the latter is naturally induced by the proposed Curriculum Dropout. After recapping the definition of Curriculum Learning, we will detail how our approach naturally induces a curriculum distribution.

Definition of curriculum distribution. Within a classical machine learning algorithm, all training examples are presented to the model in an unordered manner, frequently applying a random shuffling. Actually, this is very different from what happens for the human training process, that is education. Indeed, the latter is highly structured so that the level of difficulty of the concepts to learn is proportional to the *age* of the people, managing easier knowledge when babies and harder when adults. This "start small" paradigm will likely guide the learning process [9].

Following the same intuition, [9] proposes to subdivide the training examples based on their difficulty. Then, the learning is configured so that easier examples come first, eventually complicating them and processing the hardest ones at the end of the training. This concept is formalized by introducing a learning time $\lambda \in [0, 1]$, so that training begins at $\lambda = 0$ and ends at $\lambda = 1$. At time λ, $Q_\lambda(z)$ denotes the distribution which a generic training example z is drawn from. The notion of curriculum learning is formalized requiring that

Q_λ ensures a sampling of examples z which are easier than the ones sampled from $Q_{\lambda+\varepsilon}$, $\varepsilon > 0$. Mathematically, this is formalized by assuming

$$Q_\lambda(z) \propto W_\lambda(z)P(z). \tag{A.20}$$

In (A.20), $P(z)$ is the target training distribution, accounting for all examples, both easy and hard ones. The sampling from P is corrected by the factor $0 \leq W_\lambda(z) \leq 1$ for any λ and z. The interpretation for $W_\lambda(z)$ is the measure of the difficulty of the training example z. The maximal complexity for a training example is fixed to 1 and reached at the end of the training, *i.e.*$W_1(z) = 1$, *i.e.*$Q_1(z) = P(z)$. The weights $W_\lambda(z)$ must be chosen in such a way that

$$H(Q_\lambda) < H(Q_{\lambda+\varepsilon}), \tag{A.21}$$

where Shannon's entropy $H(Q_\lambda)$ models the fact that the quantity of information exploited by the model during training increases with respect to λ.

Curriculum dropout naturally induces a curriculum distribution. As clarified in the paper, let us consider dropout applied on the input layer only. In addition to make our analysis more understandable (see Fig. **??**), this is not restrictive since we can apply the same arguments to any of the intermediate layer.

Let us denote $\mathcal{Z}_0$ the original dataset and assume to sample from it a d-dimensional image z_0 according to a distribution π. Clearly, the natural choice for π will be a uniform distribution. Moreover, here, we measure the dimensionality d of image by means of the total number of pixels.

While dropping out units in the input layer (*i.e.*pixels in z_0), we augment $\mathcal{Z}_0$ by adding all images in $\mathcal{Z}_0$ with *one* pixel se to zero (colored in black) and also all images in $\mathcal{Z}_0$ with *two* pixel se to zero and so on. This creates

the dataset $\mathcal{Z}$, effectively used for dropout training, where any image $z \in \mathcal{Z}$ is obtained from an image $z_0 \in \mathcal{Z}_0$ by corrupting it through multiplicative Bernoulli noise. Equivalently, we can think about entrywise multiplying z_0 with a binary mask b. Therefore, we get

$$\mathbb{P}[\text{sampling } z] = \mathbb{P}[\text{sampling } z_0] \cdot \mathbb{P}[\text{sampling } b] = \pi(z_0) \cdot \mathbb{P}[\text{sampling } b]$$

In other words, any dropped out image z is uniquely determined by the original image z_0 and the binary mask b. One way to characterize that masks is by counting i, that is the number of zero entries of b. That leads to

$$\mathbb{P}[\text{sampling } z] = \pi(z_0) \cdot d$$

(1 - θ)$^i \theta^{d-i}$(A.22)

since b has entries set to zero (each realized with probability $1 - \theta$) and the remaining set to one. The latter, being $d - i$ in total, are realized in correspondence of a success for the Bernoulli(θ) variable: therefore we obtain the term θ^{d-i}.

Let us introduce our curriculum function $\theta(t) = (1 - \overline{\theta}) \exp(-\gamma t) + \overline{\theta}$ (we will omit the pedix "$_{\text{curriculum}}$" for notational simplicity). Let us re-parametrize $t = \lambda T$ such that the training time (measured from 0 to the total number T of gradients updates) spans the range $[0, 1]$, starting at time $\lambda = 0$ and ending at time $\lambda = 1$. Therefore, by modifying (A.22), we introduce the following curriculum learning distribution

$$Q_\lambda(z) = \pi(z_0) \cdot d$$

(1 - $\theta(\lambda T)$)$^i \theta(\lambda T)^{d-i}$.(A.23)Let us define

$$P(z) = Q_1(z). \tag{A.24}$$

When re-parametrizing $Q_\lambda(z) = Q_\lambda(z_0, i)$, we get a mixed distribution (discrete with respect to i and continous with respect to z_0). Hence,

$$\int Q_\lambda(z)dz = \int_{\mathcal{Z}_0} \pi(z_0)dz_0 \cdot \sum_{i=0}^{d} d$$

(1 - $\theta(\lambda T))^i \theta(\lambda T)^{d-i} = 1$(A.25)

because π is a normalized over its support $\mathcal{Z}_0$ and because the second factor equals one thanks to the Binomial Theorem.

If we compute the entropy of Q_λ, we obtain

$$H(Q_\lambda) = H(\text{Binomial}(d,\theta(\lambda T))) \cdot H(\pi), \tag{A.26}$$

being

$$H(\text{Binomial}(d,\theta(\lambda T))) = \frac{1}{2}\log[2\pi e d \cdot \theta(\lambda T)(1-\theta(\lambda T)] + O\left(\frac{1}{d}\right) \tag{A.27}$$

a strictly increasing function of λ. To see that, notice that it is enough to prove that $\theta(\lambda T)(1-\theta(\lambda T))$ is increasing as a function of λ. But, this is true since composition of the composition of strictly decreasing functions is strictly increasing. Precisely, the two functions to be composed are $\theta(\lambda T)$ and $f(x) = x(1-x)$, both of them strictly decreasing. Indeed,

$$\theta'(\lambda T) = -\gamma T(1-\overline{\theta})\exp(-\gamma\lambda T) < 0$$

for any λ and

$$f'(x) = 1 - 2x < 0$$

since we evaluate $f(\theta(\lambda T))$ and $\theta(\lambda T) > \overline{\theta} \geq 1/2$ for any λ. Therefore, for any $\varepsilon > 0$,

$$H(Q_\lambda) < H(Q_{\lambda+\varepsilon}).$$

This completes the proof.

A.2 Experimental setup

In this section, we detail the network architectures and hyperparameters used for the experiments, and provide some more extensive results.

A.2.1 Gamma

As claimed in footnote 2, we here show that Curriculum Dropout with exponential scheduling, is very robust against the choice of the decaying factor γ. Namely, *any curriculum always leads to better generalization than the no-curriculum strategy* (*i.e.*the standard dropout scheme). Moreover, the heuristic $\gamma = 10/T$ defined in Chapter 3 is proved to be an effective rule. Results are reported in Table A.1 for the Cifar-10 dataset. The architecture and experimental setup are as in section A.2.2.

Dataset	Dropout	$\gamma = 10^{-3}$	$\gamma = 7 \times 10^{-4}$	$\gamma = 3 \times 10^{-4}$	$\gamma = 10^{-4}$
CIFAR-10	73.29	73.52	73.87	**73.99**	73.69

Table A.1. Accuracies on CIFAR-10 with regular Dropout and Curriculum Dropout with different values of the decaying factor γ. Best result (bold), corresponds to the heuristic proposed in the paper, *i.e.*$\gamma = 10/T$. In fact, we train the network on 50000 samples for 80 epochs, with batch-size of 128. This corresponds to $T \approx 3.1 \times 10^4$ iterations and yields $\gamma T \approx 10$.

A.2.2 Network Architectures

Tables A.2, A.3, A.4 show the network architectures used for different experiments. Relu Learning rate and momentum were set to 0.0001 (0.001 for experiments in CIFAR-10 and CIFAR-100) and 0.95, respectively, for each run. Gamma was set following the heuristics reported in the manuscript. We initialized weights with normals with std-dev 0.01. Since all architectures

Layer Type	Layer Size	Filter Size	Padding/Stride
conv	64 filters	3x3	1/1
max pool		2x2	0/2
conv	128 filters	3x3	1/1
max pool		2x2	0/2
fc	1024 units		
fc	1024 units		
softmax	# of classes		

Table A.2. Network architecture for CIFAR-10 and CIFAR-100. The only difference is the size of the softmax layer.

rely on ReLU activations, units are also initialized with a small positive bias $b = 0.01$ in order to avoid dead neurons.

A.2.3 Full Results

Table A.5 is an extended version of the one in the manuscript, in that it shows mean accuracies together with standard deviations. To calculate the mean accuracies, we selected the 10 highest accuracy values for each of the 10 runs, calculated the average of these values and then calculated the average (and the standard deviation) over the 10 mean values.

Layer Type	Layer Size	Filter Size	Padding/Stride
conv	96 filters	5x5	2/1
max pool		3x3	0/2
conv	128 filters	5x5	2/1
max pool		3x3	0/2
conv	256 filters	5x5	2/1
max pool		3x3	0/2
fc	2048 units		
fc	2048 units		
softmax	# of classes		

Table A.3. Network architecture for SVHN, Cifar, Caltech-101 and Caltech-256. The only difference is the size of the softmax layer.

Layer Type	Layer Size	Filter Size	Padding/Stride
conv	32 filters	5x5	1/1
max pool		2x2	0/2
conv	48 filters	5x5	1/1
max pool		2x2	0/2
fc	2048 units		
fc	1024 units		
softmax	10 units		

Table A.4: Network architecture for MNIST.

Dataset	Architecture	Configuration (n or $n\overline{\theta}$ fixed)	Classes	Unregularized network	Dropout [60, 142]	Anti-Curriculum	*Curriculum Dropout*
MNIST [82]	MLP	n	10	98.67 ± 0.06	99.05 ± 0.03	98.76 ± 0.04	99.02 ± 0.00
	CNN-1	n		99.25 ± 0.03	99.39 ± 0.01	99.20 ± 0.05	99.43 ± 0.01
Double MNIST	CNN-2	n	55	92.48 ± 0.82	93.91 ± 0.68	93.21 ± 0.80	94.83 ± 0.50
	CNN-2	$n\overline{\theta}$			93.36 ± 0.73	93.01 ± 0.56	93.60 ± 0.82
SVHN [111]	CNN-2	n	10	84.63 ± 0.40	86.98 ± 0.16	85.80 ± 0.16	87.28 ± 0.20
	CNN-2	$n\overline{\theta}$			86.22 ± 0.22	86.14 ± 0.15	86.69 ± 0.19
CIFAR-10 [78]	CNN-1	n	10	73.06 ± 0.15	73.29 ± 0.33	72.38 ± 0.24	73.69 ± 0.28
CIFAR-100 [78]	CNN-1	n	100	39.70 ± 0.21	40.71 ± 0.05	39.70 ± 0.24	41.36 ± 0.32
Caltech-101 [35]	CNN-2	n	101	28.56 ± 0.68	32.78 ± 0.87	30.13 ± 1.31	33.28 ± 0.77
Caltech-256 [50]	CNN-2	n	256	14.39 ± 0.64	16.75 ± 0.42	14.18 ± 0.24	17.62 ± 0.98

Table A.5: Comparison of the proposed scheduling versus [60, 142].

Appendix B

Supplementary Material for Chapter 4

B.1 Architectures

We provide in this section a detailed description of the networks used for our experiments. For the digit datasets, the encoders follow the standard architectures commonly used in unsupervised domain adaptation [39].

Figure B.1, *left*: architectures of E_S and E_I used for MNIST $\leftrightarrow$ USPS and SVHN $\rightarrow$ MNIST.

Figure B.1, *right*: architectures of E_S and E_I used for SYN $\rightarrow$ SVHN.

Figure B.2, *left*: architecture of S used for all the experiments.

Figure B.2, *right*: architecture of D_1 used for all the experiments.

Figure B.3, *left*: architecture of D_2 used for SVHN $\rightarrow$ MNIST and SYN $\rightarrow$ SVHN.

Figure B.3, *right*: architecture of D_2 used for MNIST $\leftrightarrow$ USPS and NYUD (RGB $\rightarrow$ D).

Concerning E_S and E_I used in the NYUD experiment, we relied on a pre-trained VGG-16 [138], following the protocol used by Tzeng et al. [153]. We cut it at *fc7*, which was shrieked to be 128-dim and modified with tanh activations. The classifier C consists in an additional 19-dimensional softmax layer.

Summarizing, we found out that D_2 should be built with two or three hidden layers to stabilize the minimax game against E_I (whose structure must be the same as E_S). We designed an S that proved to be reliable in all experiments; to play a balanced minimax game, we found out that a one-hidden-layer neural network as a discriminator (D_1) is an optimal choice. The size of the hidden layer depends on the problem, and can be determined by observing the stability of the training procedure.

B.2 Hyperparameters

We report in this section the hyperparameters used in the different Steps of the training procedures. Note that hyperparameters were set in order to reach the convergence of the GAN [45] minimax games, no cross-validation using target labels was performed.

B.2.1 Digits

For each training Step, we used a batch size of 64 samples. The learning rate was set to $3 \cdot 10^{-4}$ for Step 0, $1 \cdot 10^{-4}$ for Step 1 and $3 \cdot 10^{-5}$ for Step 2, in all experiments except MNIST $\leftrightarrow$ USPS, where was set to $3 \cdot 10^{-6}$.

B.2.2 NYUD

In Step 0, the network is not trained from scratches: following the protocol described in [153], we fully fine-tune a VGG-16 network [138] (pre-trained on ImageNet [27]) for 20.000 iterations, in order to have a comparable baseline model. Batch size is 32 (instead of 128) due to hardware limitations. The learning rate were 10^{-4} for Step 0, 10^{-5} for Step 1 and 10^{-7} for Step 2.

B.3 Ablation study

In the ablation study presented in the paper, we evaluate *DI LS-ADDA*, short for Domain Invariant Least Squares ADDA, *i.e.*our method without performing feature augmentation through S. Figure B.4 depicts the two Steps of such algorithm. The architectures of the modules and hyperparameters are the same as in the full pipeline.

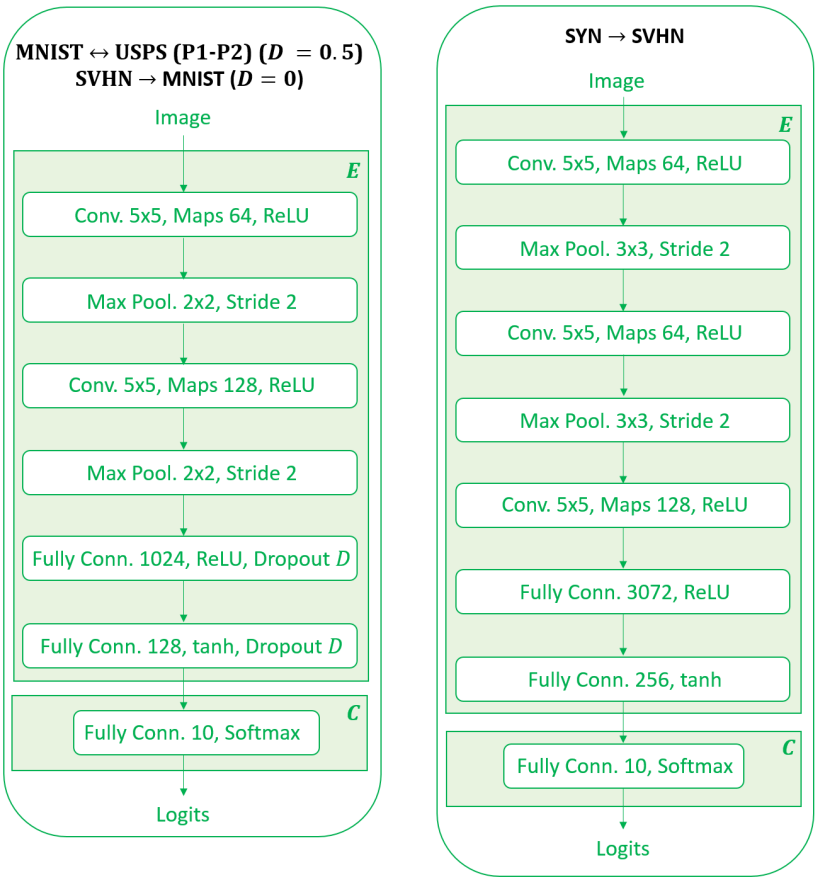


Figure B.1. Architectures used for $C \circ E_S$ and $C \circ E_I$ ($C \circ E$ for simplicity) in the MNIST ↔ USPS (P1-P2) and in the SVHN → MNIST (*left*) experiments, with the different values of Dropout [141] indicated (D), and in the SYN → SVHN experiment (*right*). The classification module (C) is a simple fully-connected + softmax layer.

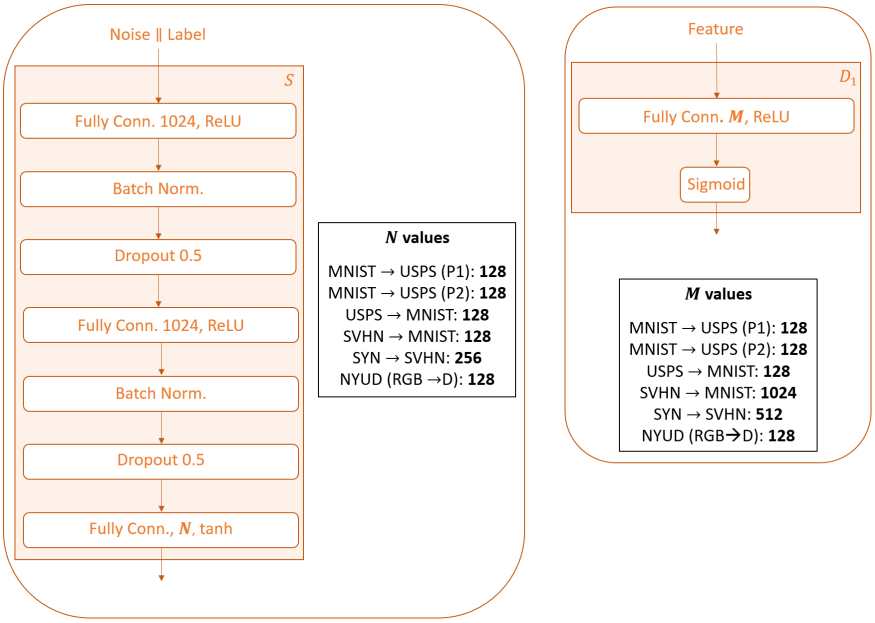


Figure B.2. Architectures used for S (*left*) and for D_1 (*right*), with the size of the features generated and of the hidden layer indicated, respectively.

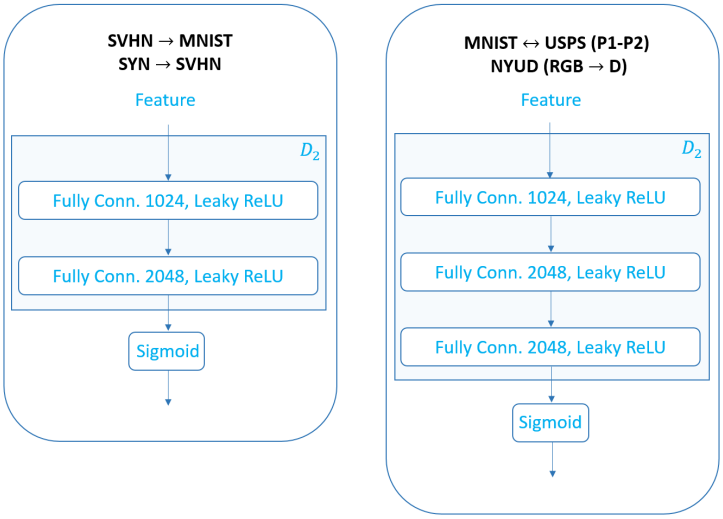


Figure B.3. Architectures used for D_2 in SVHN → MNIST and SYN → SVHN (*left*), and in NYUD and MNIST ↔ USPS (*right*).

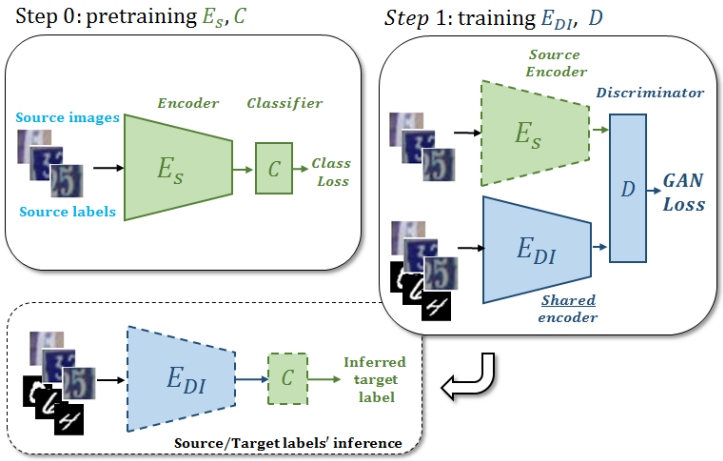


Figure B.4. *DI LS-ADDA*. Domain invariance is enforced by feeding both target and source data to E_{DI}. The feature augmentation module S is removed from the full pipeline. E_{DI} has the same architecture as E_I and D of D_2.

Appendix C

Supplementary Material for Chapter 5

C.1 Architectures and Hyperparameters

We provide in this section a detailed description of the networks used for our experiments.

Figure C.1, Figure C.3 and Figure C.2 report details regarding the architectures used for C (classifier), D (GAN's discriminator) and G (GAN's generator) in the different benchmark experiments.

We report in the following the hyperparameters associated with the same experiments. We use Adam optimizer [75] in all the experiments, and set the learning rate to train pre-train C on data from the source distribution to $3 \cdot 10^{-4}$. For the cGAN pre-training, we set the learning rate for training both G and D to 10^{-5}. When running Algorithm 2, we set $\eta = 5 \cdot 10^{-5}$ and $\delta = 5 \cdot 10^{-5}$.

Architectural choices, as well as hyperparameter tuning, were carried out with the goal of making GANs converge.

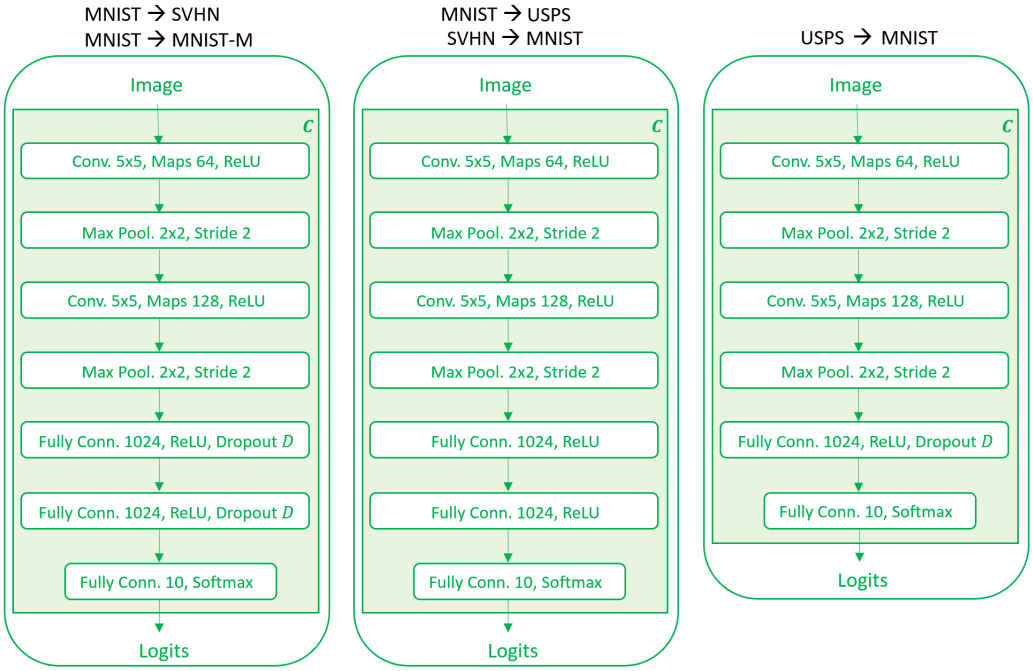


Figure C.1: Architecture for the classifier C (see Figure 5.3).

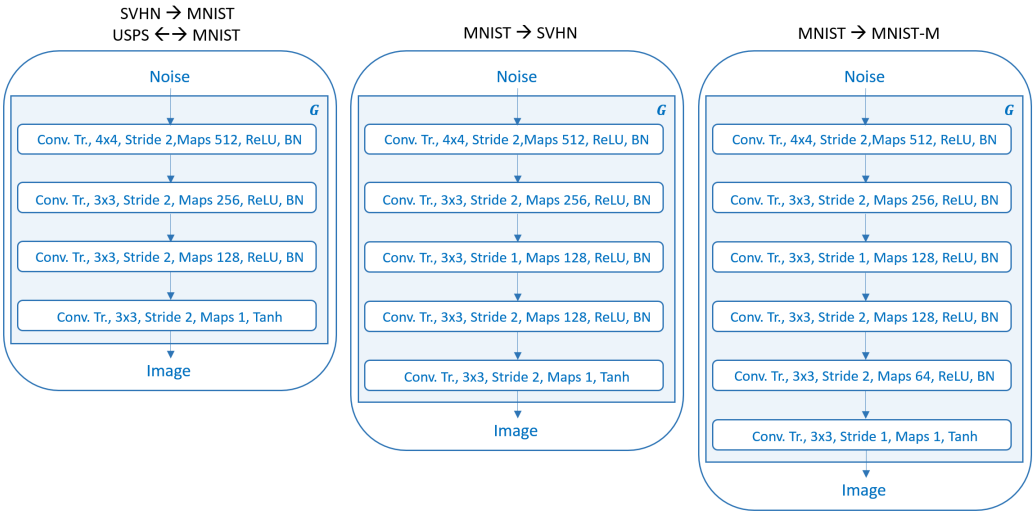


Figure C.2: Architecture for the generator G (see Figure 5.3).

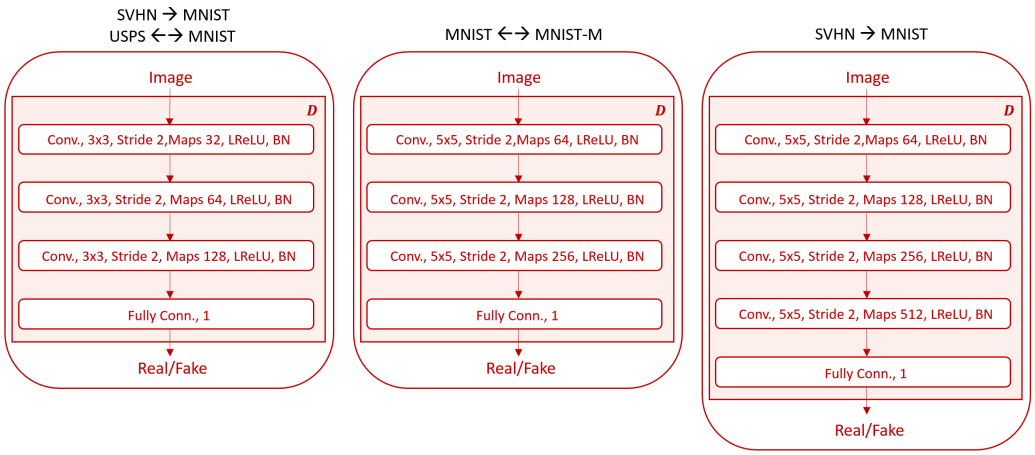


Figure C.3: Architecture for the discriminator D (see Figure 5.3).

Appendix D

Supplementary Material for Chapter 6

D.1 Proofs

In this section, we provide the proofs for the statements reported in Chapter 6[1].

D.1.1 Proof of Theorem 1

Recall that we consider a fixed $\theta \in \Theta$, $x_0 \in \mathcal{X}$, $y_0 \in \mathcal{Y}$, and $z_0 = g(\theta_f; x_0)$. We begin by noting that since $\mathrm{Im}(g(\theta_f; \cdot)) = \mathbb{R}^p$, we have

$$\begin{aligned}\phi_\gamma(\theta; (x_0, y_0)) &= \sup_{x \in \mathcal{X}} \{\ell(\theta; (x, y_0)) - \gamma c_\theta((x, y_0), (x_0, y_0))\} \\ &= \sup_{z \in \mathbb{R}^p} \left\{\ell(\theta; (z, y_0)) - \frac{\gamma}{2} \|z - z_0\|_2^2 =: h(z)\right\}. \end{aligned} \tag{D.1}$$

Similarly as $x_\epsilon^\star$, let $z_\epsilon^\star$ be an ϵ-optimizer to the problem (D.1)

$$z_\epsilon^\star \in \epsilon\text{-}\arg\max_{z \in \mathbb{R}^p} \left\{\ell(\theta; (z, y_0)) - \frac{\gamma}{2} \|z - z_0\|_2^2\right\}.$$

[1]Collaboration with Hongseok Namnoong, co-first author of the paper *Generalizing to Unseen Domains via Adversarial Data Augmentation* [159].

To further ease notation, let us denote

$$\begin{aligned}\ell_1(\theta;(z,y_0)) &:= \ell(\theta;(z_0,y_0)) + \nabla_z \ell(\theta;(z_0,y_0))^\top (z-z_0)\\ \ell_2(\theta;(z,y_0)) &:= \ell(\theta;(z_0,y_0)) + \nabla_z \ell(\theta;(z_0,y_0))^\top (z-z_0)\\ &\quad + \frac{1}{2}(z-z_0)^\top \nabla_{zz}\ell(\theta;(z_0,y_0))(z-z_0),\end{aligned}$$

the first- and second-order approximation of $z \mapsto \ell(\theta;(z,y_0))$ around $z = z_0$ respectively.

First, we note that $\|\nabla_z z \ell(\theta;(z,y))\| \leq L < \gamma$ by hypothesis and hence, $\widehat{g}_{\text{newton}}(\theta_f; x_0)$ attains the maximum in the problem (D.1)

$$\begin{aligned}\widehat{g}_{\text{newton}}(\theta_f; x_0) &= z_0 + \frac{1}{\gamma}\left(I - \frac{1}{\gamma}\nabla_{zz}\ell(\theta;(z_0,y_0))\right)^{-1} \nabla_z \ell(\theta;(z_0,y_0)) \\ &= \arg\max_{z\in\mathbb{R}^p} \left\{ \ell_2(\theta;(z,y_0)) - \frac{\gamma}{2}\|z-z_0\|_2^2 := h_2(z) \right\}\end{aligned} \tag{D.2}$$

Now, note that $h_2(z) = \ell_2(\theta;(z,y_0)) - \frac{\gamma}{2}\|z-z_0\|_2^2$ is $(\gamma - L_1)$ - strongly concave since

$$\lambda_{\max}(\nabla_z z h(z)) \leq \lambda_{\max}(\nabla_z z \ell_2(\theta;(z,y_0))) + \gamma \leq \gamma - L_1$$

by Assumption 1, where $\lambda_{\max}$ denotes the maximum eigenvalue. Recalling the definition of $h(z)$ given in Eq (D.1), we then have

$$\begin{aligned}\frac{\gamma - L_1}{2}\|z_\epsilon^\star - \widehat{g}_{\text{newton}}(\theta_f; x_0)\| &\leq h_2(z_\epsilon^\star) - h_2(\widehat{g}_{\text{newton}}(\theta_f; x_0)) \\ &= h(z_\epsilon^\star) - h(\widehat{g}_{\text{newton}}(\theta_f; x_0)) + h_2(z_\epsilon^\star) - h(z_\epsilon^\star) \\ &\quad + h(\widehat{g}_{\text{newton}}(\theta_f; x_0)) - h_2(\widehat{g}_{\text{newton}}(\theta_f; x_0)) \\ &\leq \epsilon + h_2(z_\epsilon^\star) - h(z_\epsilon^\star) \\ &\quad + h(\widehat{g}_{\text{newton}}(\theta_f; x_0)) - h_2(\widehat{g}_{\text{newton}}(\theta_f; x_0))\end{aligned} \tag{D.3}$$

where we used the definition of $z_\epsilon^\star$ in the last inequality.

Next, we note that h_2 and h are close by Taylor expansion.

Lemma 2 ([110, Lemma 1]). *Let $f : \mathbb{R}^p \to \mathbb{R}$ have a L-Lipschitz Hessian so that for all $z, z' \in \mathbb{R}^p$, $\|\nabla_z z f(z) - \nabla_z z f(z')\| \leq L \|z - z'\|_2$. Then, for all $z, z' \in \mathbb{R}^p$,*

$$\left| f(z') - f(z) - \nabla f(z)^\top (z' - z) - \frac{1}{2}(z' - z)^\top \nabla_z z f(z)(z' - z) \right| \leq \frac{L}{6} \|z' - z\|_2^2 .$$

Applying Lemma 2, we have that

$$|h_2(z) - h(z)| \leq \frac{L_2}{6} \|z - z_0\|_2^3 .$$

Using this inequality in the bound (D.3), we arrive at

$$\begin{aligned} &\frac{\gamma - L_1}{2} \|z_\epsilon^\star - \widehat{g}_{\text{newton}}(\theta_f; x_0)\|_2 \\ &\leq \epsilon + \frac{L_2}{6} \left(\|z_0 - z\top_\epsilon\|_2^3 + \|z_0 - \widehat{g}_{\text{newton}}(\theta_f; x_0)\|_2^3 \right) \end{aligned} \tag{D.4}$$

From definition (D.2) of $\widehat{g}_{\text{newton}}(\theta_f; x_0)$, we have

$$\|z_0 - \widehat{g}_{\text{newton}}(\theta_f; x_0)\|_2^3 \leq \left(\frac{1}{\gamma}\right)^3 \left(\frac{\gamma}{\gamma - L_1}\right)^3 L_0^3. \tag{D.5}$$

Next, to bound $\|z_0 - z\top_\epsilon\|_2$ in the bound (D.4), we show that $z_\epsilon^\star$ and z_0 are at most $O(1/\gamma)$-away. We defer the proof of the following lemma to Appendix D.1.2

Lemma 3. *Let Assumption 1 hold and $Im(g(\theta_f; \cdot)) = \mathbb{R}^p$. Then,*

$$\left\| z_\epsilon^\star - z_0 - \frac{1}{\gamma} \nabla_z \ell(\theta; (z_0, y_0)) \right\|_2 \leq \frac{4L_0}{\gamma} + \sqrt{\frac{2\epsilon}{\gamma}}.$$

Applying Lemma 3 to bound $\|z_0 - z\top_\epsilon\|_2^3$ on the right hand side of inequality (D.4), and using the bound (D.5) for $\|z_0 - \widehat{g}_{\text{newton}}(\theta_f; x_0)\|_2^3$, we obtain

$$\frac{\gamma - L_1}{2} \|z_\epsilon^\star - \widehat{g}_{\text{newton}}(\theta_f; x_0)\|_2 \leq \epsilon + \frac{L_2}{6} \left[4 \left(\frac{5L_0}{\gamma}\right)^3 + \left(\frac{2\epsilon}{\gamma}\right)^{\frac{3}{2}} + \left(\frac{L_0}{\gamma - L_1}\right)^3 \right].$$

This gives the final result.

D.1.2 Proof of Lemma 3

We use the following key lemma which says that for functions that satisfy a growth condition, its minimum is stable to perturbations to the function.

Lemma 4 ([15, Proposition 4.32])**.** *Suppose that f_0 satisfies the second-order growth condition: there exists a $c > 0$ such that if we denote by $z^\star$ the minimizer of f so that $f_0(z^\star) = \inf_{z\in\mathbb{R}^p} f_0(z)$, we have for all z*

$$f_0(z) \geq f_0(z^\star) + c\left\|z - z^\star\right\|_2^2.$$

If there is a function $f_1 : \mathbb{R}^p \to \mathbb{R}$ such that $f_0 - f_1$ is κ-Lipschitz on a neighborhood N of $x^\star$, then z, any ϵ-approximate minimizer of f_1 in N, satisfies

$$\left\|z - z^\star\right\|_2 \leq c^{-1}\kappa + c^{-1/2}\epsilon^{1/2}$$

Letting $f_0(z) := -\ell_1(\theta;(z,y_0)) + \frac{\gamma}{2}\left\|z - z_0\right\|_2^2$ and $f_1(z) := -h(z) = -\ell(\theta;(z,y_0)) + \frac{\gamma}{2}\left\|z - z_0\right\|_2^2$, note first that f_0 is $\frac{\gamma}{2}$ strongly convex. Further, $f_0(z) - f_1(z) = \ell(\theta;(z,y_0)) - \ell_1(\theta;(z,y_0))$ is $2L_0$-Lipschitz by Assumption 1. Applying Lemma 4, we obtain the result.

D.1.3 Proof of Theorem 2

Again, we abuse notation by writing $\ell(\theta;(z,y)) = \ell(\theta;(x,y))$ for $z = g(\theta_f;x) \in \mathbb{R}^p$, and similarly $p_j(\theta;z)$ and $\phi_\gamma(\theta;z)$. We begin by noting that since $\mathrm{Im}(g(\theta,\cdot)) = \mathbb{R}^p$, we have

$$\phi_\gamma(\theta;(x,y)) = \sup_{z'\in\mathbb{R}^p}\left\{\ell(\theta;(z',y)) - \frac{\gamma}{2}\left\|z - z'\right\|_2^2\right\}.$$

The following claim will be crucial.

Claim 5. *If $z \mapsto \nabla_z\ell(\theta;(z,y))$ is L-Lipschitz with respect to the $\left\|\cdot\right\|_2$-norm, then*

$$\frac{1}{\gamma+L}\left\|\nabla_z\ell(\theta;(z,y))\right\|_2^2 \leq \phi_\gamma(\theta;(z,y)) - \ell(\theta;(z,y)) \leq \frac{1}{\gamma-L}\left\|\nabla_z\ell(\theta;(z,y))\right\|_2^2.$$

Proof of Claim From Taylor's theorem, we have

$$\left|\ell(\theta;(z',y)) - \ell(\theta;(z,y)) - \nabla_z \ell(\theta;(z,y))^\top (z-z')\right| \leq \frac{1}{2} L \left\|z-z'\right\|_2^2 .$$

Using this approximation in the definition of $\phi_\gamma(\theta;(z,y))$, we get

$$\begin{aligned}\phi_\gamma(\theta;(z,y)) &\leq \sup_{z'} \left\{ \ell(\theta;(z,y)) + \nabla_z \ell(\theta;(z,y))^\top (z-z') - \frac{\gamma - L}{2} \left\|z-z'\right\|_2^2 \right\} \\ &= \ell(\theta;(z,y)) + \frac{1}{2(\gamma - L)} \left\|\nabla_z \ell(\theta;(z,y))\right\|_2^2 .\end{aligned}$$

Similarly, we can compute the lower bound

$$\begin{aligned}\phi_\gamma(\theta;(z,y)) &\geq \sup_{z'} \left\{ \ell(\theta;(z,y)) + \nabla_z \ell(\theta;(z,y))^\top (z-z') - \frac{\gamma + L}{2} \left\|z-z'\right\|_2^2 \right\} \\ &= \ell(\theta;(z,y)) + \frac{1}{2(\gamma + L)} \left\|\nabla_z \ell(\theta;(z,y))\right\|_2^2 .\end{aligned}$$

Combining the two bounds, the claim follows. □

From the claim, it suffices to show that $z \mapsto \nabla_z \ell(\theta;(z,y))$ is L-Lipschitz. From $\nabla_z \ell(\theta;(z,y)) = -\theta_{c,y} + \sum_{j=1}^m p_j(\theta;z)\theta_{c,j}$, we have

$$\left\|\nabla_z \ell(\theta;(z',y)) - \nabla_z \ell(\theta;(z,y))\right\|_2 = \left\| \sum_{j=1}^m (p_j(\theta;z) - p_j(\theta;z'))\theta_j \right\|_2 .$$

Now, since

$$\left\|\nabla_z p_j(\theta;z)\right\|_2 = \left\| -p_j(\theta;z) \left(\theta_j - \sum_{l=1}^m p_l(\theta;z)\theta_l \right) \right\|_2 \leq 2 \max_{1\leq j\leq m} \left\|\theta_{c,j}\right\|_2 ,$$

we conclude that

$$\left\|\nabla_z \ell(\theta;(z',y)) - \nabla_z \ell(\theta;(z,y))\right\|_2 \leq L \left\|z-z'\right\|_2 .$$